MODUL

MIKROPROSESOR DAN SISTEM ANTARMUKA

OLEH :
NARDI, S.T, M.Kom
HARYANTO, MT

PROGRAM STUDI
INSTRUMENTASI METEOROLOGI KLIMATOLOGI DAN GEOFISIKA
SEKOLAH TINGGI METEOROLOGI KLIMATOLOGI DAN GEOFISIKA
TANGERANG SELATAN
2018

KATA PENGANTAR

Puji dan syukur kami panjatkan kehadirat Allah SWT, karena berkat segala karunia, rahmat dan hidayah-Nya sehingga pembuatan modul praktik ini dapat terselesaikan sebagai panduan untuk praktik mata kuliah Mikroprosesor dan Sistem Antarmuka.

Modul Praktik ini secara khusus ditujukan untuk taruna Sekolah Tinggi Meteorologi Klimatologi dan Geofisika (STMKG) Program Studi Instrumentasi MKG Diploma empat (D.IV) pada mata kuliah Mikroprosesor dan Sistem Antarmuka, untuk mendemonstrasikan pemrograman Bahasa Assamble dan perancangan rangkaian-rangkaian dengan menggunakan mikrokontroler yang sering ditemukan pada sistem Instrumentasi. Dengan demikian apa yang didapat dari teoritis dapat dipraktikkan secara langsung. Hal lain yang didapat taruna yaitu mampu memahami dan menganalisa dari perintah-perintah mnemonic untuk Bahasa mesin dan mampu merancang rangkaian sistem mikroprosesor.

Dalam modul ini dibahas tentang merancang rangkaian sistem mikroprosesor dan menginstalasi program-program yang digunakan dalam praktik, membuat compiler dengan program com dan exe, memahami sistem pengalamatan dan operasi aritmatika dengan debugger, membuat downloader sebagai alat pengisian program ke IC mikrokontroler, mengakses perangkat I/O di mikroprosesor, mengakses dan mengaplikasikan perangkat interupsi pada mikrokontroler, menggunakan fitur timer/counter mikrokontroler, menggunakan LCD sebagai penampilan, menggunakan ADC yang ada di dalam mikrokontroler, mengerti cara melakukan komunikasi serial dengan mikrokontroller, menggunakan mikrokontroller untuk inputan sensor digital.

Di setiap modul disampaikan bahwa taruna mampu memahami tujuan praktikum, mengetahui peralatan-peralatan yang digunakan, mampu merancang rangkaian-rangkaian dan bahasa pemrogramannya serta mempratikkan, mengambil data, menganalisa dan mampu membuat kesimpulan.

Modul ini digunakan dalam satu semester dengan harapan taruna mampu mengimplementasikan dalam aplikasi-aplikasi mikroprosesor.

Tangerang Selatan, 2018

SATUAN ACARA PERKULIAHAN

No	Pertemuan	Materi
1	Minggu 1	Penjelasan umum tentang aturan-aturan praktikum, tatacara praktikum, format laporan dan pengenalan peralatan-peralatan yang digunakan untuk praktik.
2	Minggu 2-3	Materi perancangan sistem mikroprosesor dan Instalasi Program
3	Minggu 4	Program Com dan Exe
4	Minggu 5	Pengalamatan dan Aritmatika
5	Minggu 6-7	Downloader
6	Minggu 8	UTS
7	Minggu 9	Input Output
8	Minggu 10	Interupt
9	Minggu 11	Timer dan Counter
10	Minggu 12	Liquid Cristal Display (LCD)
11	Minggu 13	Analog Digital Converter (ADC)
12	Minggu 14	Komunikasi Serial
13	Minggu 15	Input Sensor Digital DHT11

DAFTAR ISI

Modul 1
INSTALASI PROGRAM

I. TUJUAN :

a. Memahami cara menginstalasi program DOSBOX, TURBO ASSAMBLER, PROTEUR dan CVAVR.
b. Memahami fungsi dan penggunaan program yang di instal.

II. ALAT DAN BAHAN :

a. 1 Unit Personal Computer (PC) / Laptop.
b. Master Program DOSBox.
c. Master Program Turbo Assambler.
d. Program Proteus
e. Program CVAVR

III. DASAR TEORI

DOSBox

DOSBox adalah program yang mensimulasikan fungsi dari MS-DOS, termasuk suara, grafis, input, dan jaringan. Program ini digunakan untuk menjalankan permainan video lama yang dibuat khusus untuk sistem operasi MS-DOS.

TURBO ASSAMBLER

Turbo Assembler (TASM) adalah paket assembler dikembangkan oleh Borland menghasilkan kode untuk 16 – 32 – dan 64 bit x86 MS-DOS atau Microsoft Windows. Hal ini dapat digunakan dengan Bahasa pemrograman bahasa tingkat tinggi Borland compiler, seperti Turbo Pascal, Turbo Basic, Turbo C dan Turbo C + + . Paket Turbo Assembler dibundel dengan Turbo Linker, dan interoperable dengan Turbo Debugger. TASM dapat merakit dari sumber MASM menggunakan model MASM dan memiliki mode ideal dengan beberapa perangkat tambahan. Pemrograman berorientasi objek telah didukung sejak versi 3.0. Versi terbaru dari Turbo Assembler adalah 5.0, tahun 1996 dan patch sampai dengan tahun 2002, dan dapat diterapkan pada Delphi dan C + + Builder.

Bahasa Assembly adalah bahasa pemrograman tingkat rendah. Dalam pemrograman komputer dikenal dua jenis tingkatan bahasa, jenis yang pertama adalah bahasa pemrograman tingkat tinggi (high level language) dan jenis yang kedua adalah bahasa pemrograman tingkat rendah (low level language).

Bahasa pemrograman tingkat tinggi lebih berorientasi kepada manusia yaitu bagaimana agar pernyataan-pernyataan yang ada dalam program mudah ditulis dan dimengerti oleh manusia. Sedangkan bahasa tingkat rendah lebih berorientasi ke mesin, yaitu bagaimana agar komputer dapat langsung mengintepretasikan pernyataan-pernyataan program.

Kelebihan Bahasa Assembly:

1. Ketika di-compile lebih kecil ukuran.
2. Lebih efisien/hemat memori.
3. Lebih cepat dieksekusi.

Kesulitan Bahasa Assembly:

1. Dalam melakukan suatu pekerjaan, baris program relatif lebih panjang dibanding bahasa tingkat tinggi.
2. Relatif lebih sulit untuk dipahami terutama jika jumlah baris sudah terlalu banyak.

3. Lebih sulit dalam melakukan pekerjaan rumit, misalnya operasi matematis.

Proteus

Software Proteus adalah sebuah software yang digunakan untuk mendesain PCB yang juga dilengkapi dengan simulasi PSpice pada level skematik sebelum rangkaian skematik di-upgrade ke PCB untuk memastikan PCB dapat berfungsi dengan semestinya. Proteus mengkombinasikan program ISIS untuk membuat skematik desain rangkaian dengan program ARES untuk membuat layout PCB dari skematik yang dibuat, sedangkan ARES atau disebut juga Advanced Routing and Editing Software digunakan untuk membuat modul layout PCB.

Proteus sangat berguna untuk desain rangkaian mikrokontroler juga berguna untuk belajar elektronika seperti dasar-dasar elektronika sampai pada aplikasi mikrokontroler.

Fitur-fitur dari Proteus adalah sebagai berikut :

1. Memiliki kemampuan untuk mensimulasikan hasil rancangan baik digital maupun analog maupun gabungan keduanya.
2. Mendukung instrumen-instrumen virtual seperti voltmeter, ammeter, osciloscope, logic analyser, dan masih banyak lagi.
3. Memiliki model-model peripheral yang interactive seperti LED, tampilan LCD, RS232, dan berbagai jenis library lainnya.
4. Memiliki kemampuan menampilkan berbagi jenis analisis secara grafis seperti transient, frekuensi, noise, distorsi, AC dan DC, dan masih banyak lagi.
5. Mendukung simulasi berbagai jenis microcontroller.
6. Mendukung berbagai jenis komponen-komponen analog.
7. Mendukung open architecture sehingga pengguna dapat memasukkan program seperti C++/ Arduino untuk keperluan simulasi.
8. Mendukung pembuatan PCB yang di-update secara langsung dari program ISIS ke program pembuat PCB-ARES.

Fitur-Fitur dari ISIS adalah sebagai berikut :

1. Dapat dioperasikan pada Windows 98/XP/7 sampai dengan Windows terbaru.
2. Adanya fasilitas pemilihan komponen dan pemberian properties.
3. Memiliki fasilitas report terhadap kesalahan-kesalahan perancangan dan simulasi elektrik.
4. Routing secara otomatis dan memiliki fasilitas penempatan dan penghapusan dot.
5. Mendukung untuk perancangan berbagai jenis bus dan komponen-komponen pin, port modul dan jalur.
6. Mendukung fasilitas interkoneksi dengan program pembuat PCB-ARES.
7. Memiliki fasilitas untuk menambahkan package dari komponen yang belum didukung.

Fitur-fitur dari ARES adalah sebagai berikut :

1. Terintegrasi dengan program pembuat skematik ISIS, dengan kemampuan untuk menentukan informasi routing pada skematik.
2. Memiliki database dengan tingkat keakuratan 32-bit dan memberikan resolusi sampai 10 nm, resolusi angular 0,1 derajat dan ukuran maksimum board sampai kurang lebih 10 m. ARES mendukung sampai 16 layer.
3. Visualisasi board 3-Dimensi.
4. Penggambaran 2-Dimensi dengan simbol library

CVAVR :

CodeVision AVR merupakan sebuah software yang digunakan untuk memprogram mikrokontroler. Mulai dari penggunaan untuk kontrol sederhana sampai kontrol yang cukup kompleks, mikrokontroler dapat berfungsi jika telah diisi sebuah program, pengisian program ini dapat dilakukan menggunakan compiler yang selanjutnya diprogram ke dalam mikrokontroler menggunakan fasilitas yang sudah di sediakan oleh program tersebut.
CodeVision AVR mempunyai suatu keunggulan dari compiler lain, yaitu adanya codewizard, fasilitas ini memudahkan kita dalam inisialisasi mikrokontroler yang akan kita gunakan.
CodeVisionAVR pada dasarnya merupakan perangkat lunak pemrograman mikrokontroller keluarga AVR berbasis bahasa C. Ada tiga komponen penting yang telah diintegrasikan dalam perangkat lunak ini: Compiler C, IDE dan Program generator.
Khusus untuk library fungsi, disamping library standar (seperti fungsi-fungsi matematik, manipulasi String, pengaksesan memori dan sebagainya), CodeVisionAVR juga menyediakan fungsi-fungsi tambahan yang sangat bermanfaat dalam pemrograman antarmuka AVR dengan perangkat luar yang umum digunakan dalam aplikasi kontrol. Beberapa fungsi library yang penting diantaranya adalah fungsi-fungsi untuk pengaksesan LCD, komunikasi I2C, IC RTC (Real time Clock), sensor suhu LM75, SPI (Serial Peripheral Interface) dan lain sebagainya.

CodeVisionAVR adalah suatu kompiler berbasis bahasa C, yang terintegrasi untuk memprogram dan sekaligus compiler aplikasi AVR (Alf and Vegard's Risc processor) terhadap mikrokontroler dengan sistem berbasis window. CodeVisionAVR ini dapat mengimplematasikan hampir semua interuksi bahasa C yang sesuai dengan arsitektur AVR, bahkan terdapat beberapa keunggulan tambahan untuk memenuhi keunggulan spesifikasi dari CodeVisionAVR yaitu hasil kompilasi studio debugger dari ATMEL.

Disamping library standar C, CodeVisionAVR C compiler memiliki librari lain untuk:
Modul LCD Alpanumerik, Delays, Protokol semikonduktor Maxim/Dallas, dan lainnya CodeVisionAVR juga memiliki CodeWizardAVR sebagaki generator program otomatis, yang memungkinkan kita untuk menulis, segala bentuk pengaturan Chip dalam waktu singkat, dan semua kode yang dibutuhkan untuk mengimplementasikan fungsi-fungsi seperti:

1. **Pengaturan akses External Memory.** Untuk chip-chip AVR yang memungkinkan koneksi memori eksternal SRAM, dapat juga mengatur ukuran memori dan wait state (tahap tunggu) dari memori ketika memori tersebut diakses.
2. **Identifikasi chip reset source.** Adalah suatu layanan dimana kita dapat membuat kode secara otomatis yang dapat mengidentifikasi kondisi yang menyebabkan chip di reset.
3. **Inisialisasi port input/output.** Pengaturan port-port yang kan dijadikan gerbang masukan dan keluaran dapat secara otomatis digenerate codenya. Yang kita lakukan hanya memilih port-port yang akan digunakan sebagai input atau output.
4. **Inisialisasi Interupsi external.** Pengaturan interupsi eksternal yang nantinya akan digunakan untuk menginterupsi program utama
5. **Inisialisasi timers/counters.** Pengaturan timers yang berfungsi untuk mengatur frekwensi yang nantinya digunakan pada interupsi.
6. **Inisialisasi timer watchdog.** Pengaturan timers yang berfungsi untuk mengatur frekwensi yang nantinya digunakan pada interupsi, sehingga interupsi akan dilayani oleh suatu fungsi wdt_timeout_isr .

7. **Inisialisasi UART(USART) dan komunuikasi serial.** Pengaturan komunikasi serial sebagai penerima atau pengirim data.
8. **Inisialisasi komparasi analog.** Pengaturan yang berkaitan dengan masukan data yang digunakan dalam aplikasi yang membutuhkan komparasi pada ADC nya.
9. **Inisialisasi ADC.** Pengaturan ADC(Analog-Digital Converter) yang berfungsi untuk merubah format analog menjadi format digital untuk diolah lebih lanjut.
10. **Inisialisasi antarmuka SPI.** Pengaturan chip yang berkaitan dengan Clock rate, Clock Phase, dan lainnya.
11. **Inisialisasi antarmuka Two Wire BUS.** Pengaturan Chip yang berhubungan dengan pola jalur komunikasi antara register yang terdapat pada chip AVR.
12. **Inisialisasi antarmuka CAN.** Pengaturan chip yang lebih kompleks, yang dapat mengatur interupsi, transmisi data, timers, dan lainnya.
13. **Inisialisasi sensor temperatur, thermometer, dan lainnya.** Pengaturan yang berhubungan dengan sensor temperatur one wire bus, memiliki fungsi-fungsi yang ada pada librari CodeVisionAVR.
14. **Inisialisasi one wire bus.** Pengaturan yang berhubungan dengan sensor temperatur yang memiliki fungsi-fungsi yang ada pada librari CodeVisionAVR. Seperti Maxim/Dallas Semiconductor.
15. **Inisialisasi modul LCD.** Pengaturan port-port yang akan digunakan sebagai penghubung dengan LCD alphanumerik.

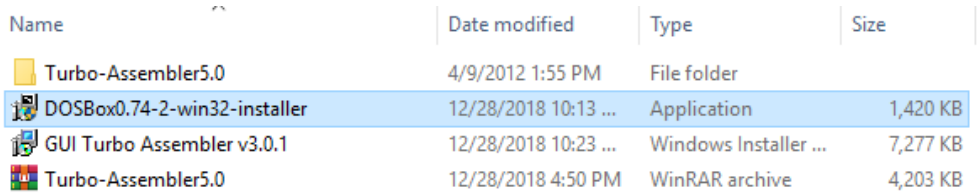
IV. TUGAS PENDAHULUAN

1. Apa yang anda ketahui tentang Turbo Assambler
2. Jelaskan dengan singkat fitur-fitur pada Program Proteus
3. Jelaskan dengan singkat fitur-fitur pada Program CVAVR.

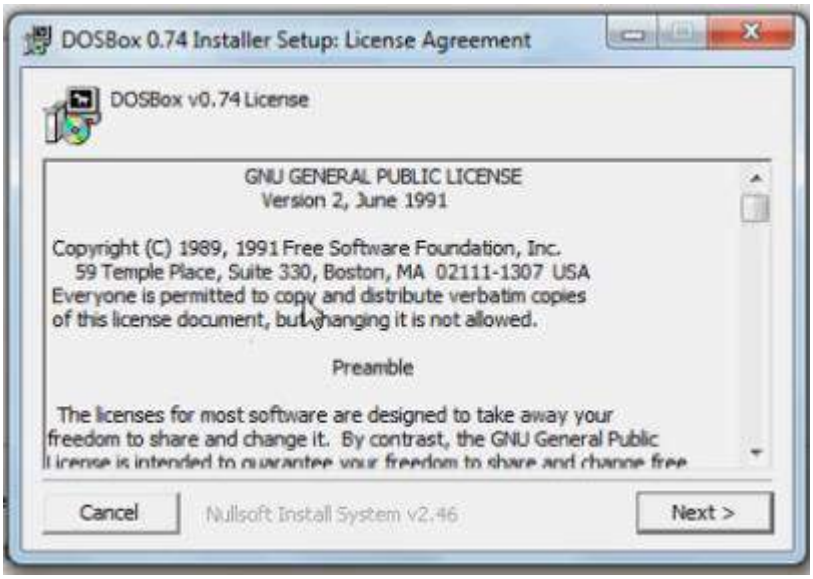
V. PROSEDUR DAN PENGAMATAN PERCOBAAN

A. Instalasi Program DOSBox

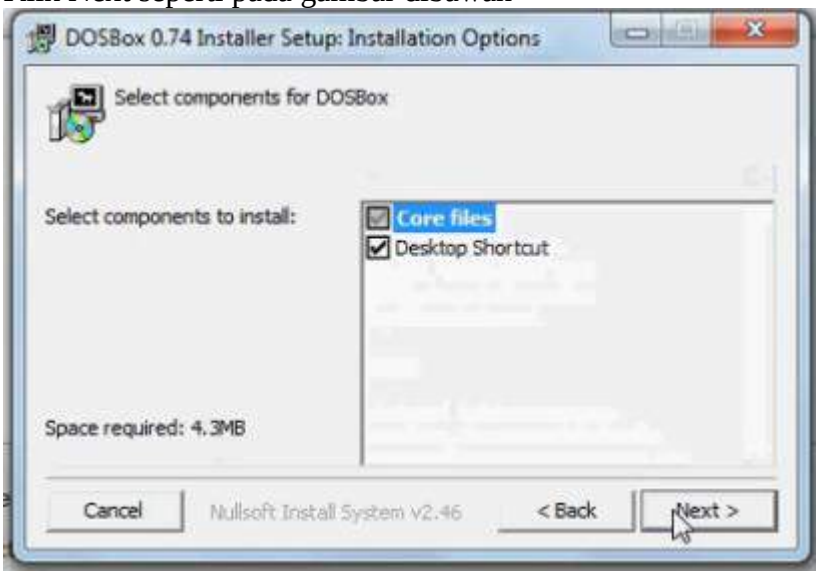
1. Siapkan Master Program DOSBox dan Turbo Assambler.
2. Klik 2 kali pada folder DOSBox0.74-2-win32-installer.



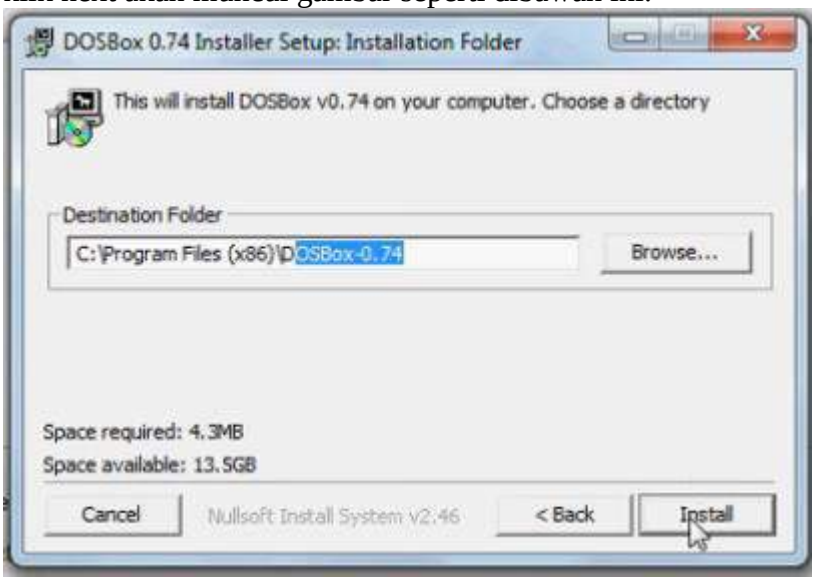
3. Klik Next seperti pada gambar dibawah.



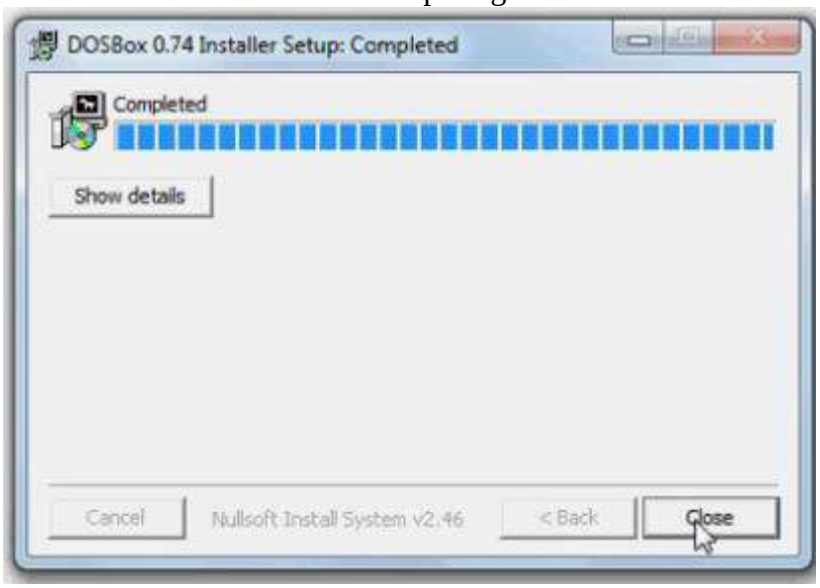
4. Klik Next seperti pada gambar dibawah



5. Centang Desktop Shortcut bila diinginkan ada di desktop Shorcut, selanjutnya klik next akan muncul gambar seperti dibawah ini.



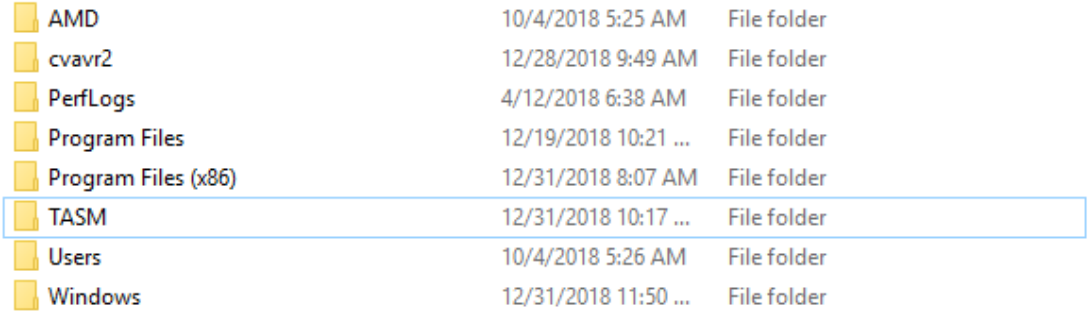
6. Klik Instal maka akan muncul seperti gambar di bawah ini



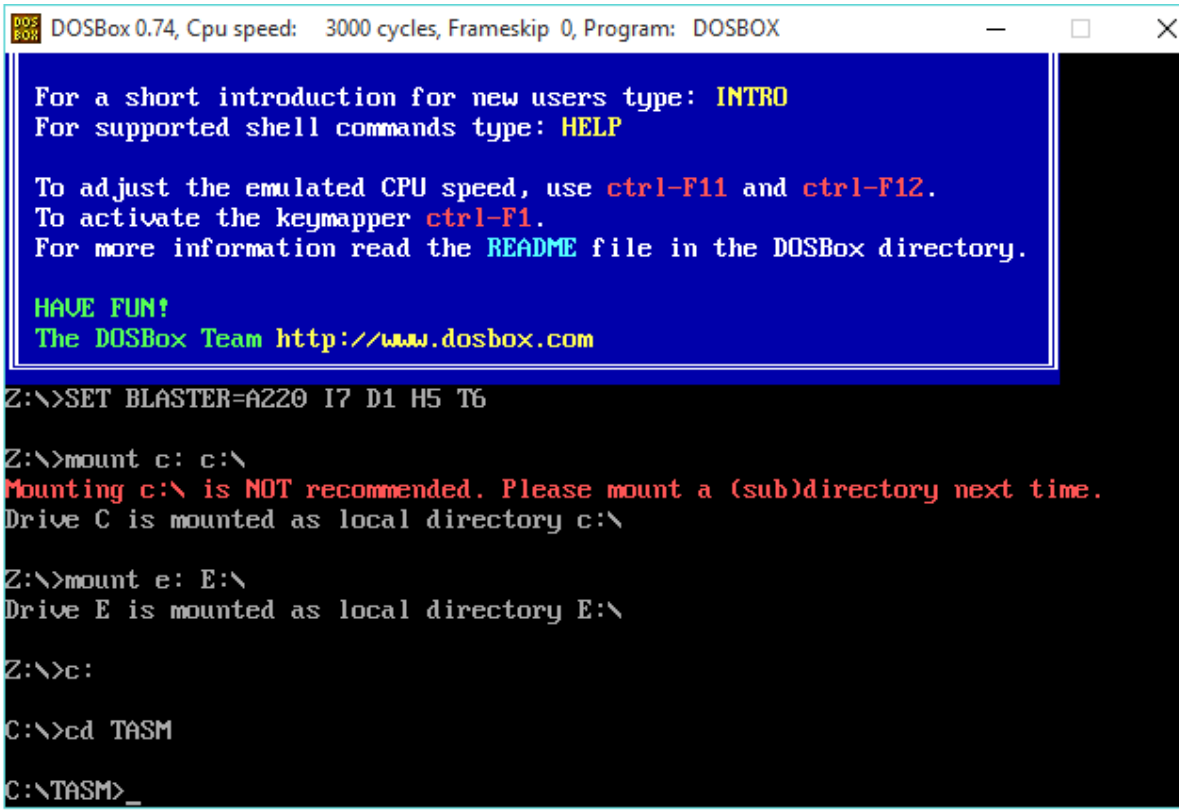
7. Tunggu sampai Installer Setup Completed, selanjutnya klik Close.

B. Instalasi Program Turbo Assambler.

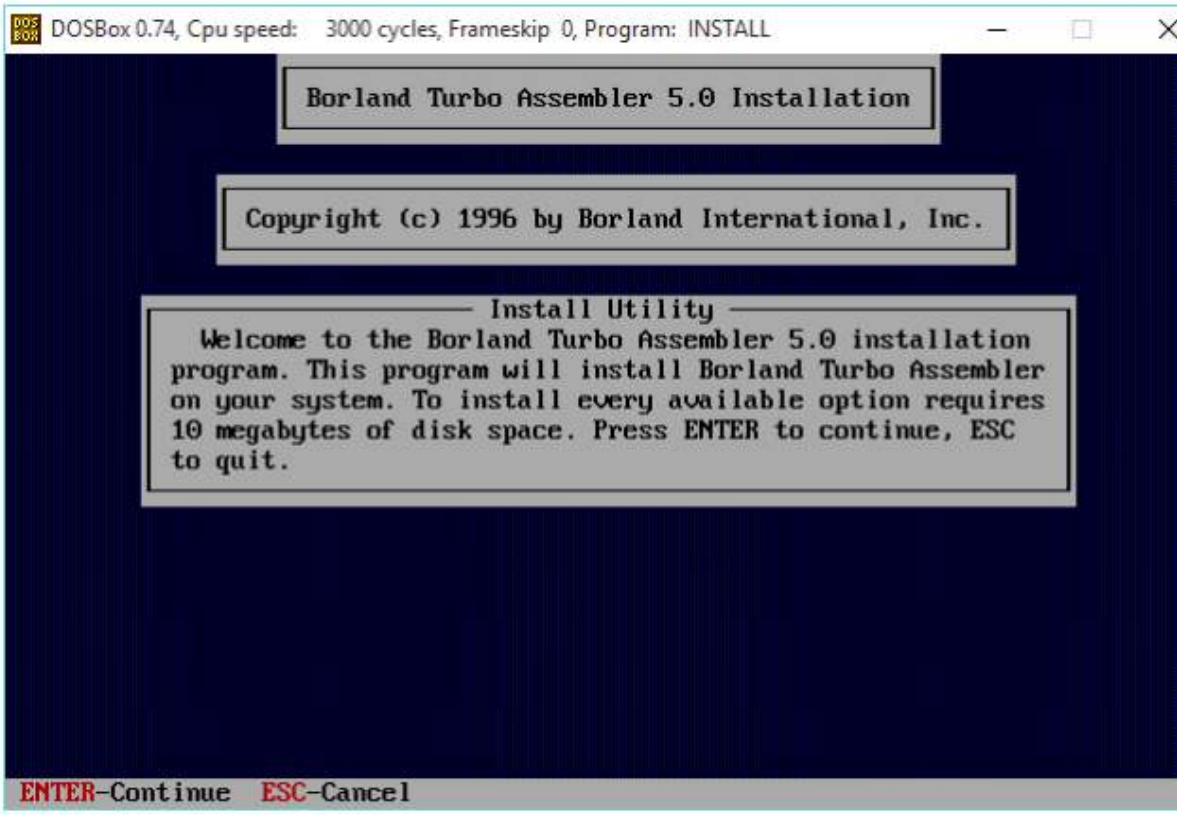
1. Buatlah sebuah folder di drive C: dan copy semua isi yang ada dalam folder Turbo Installer 5.0.



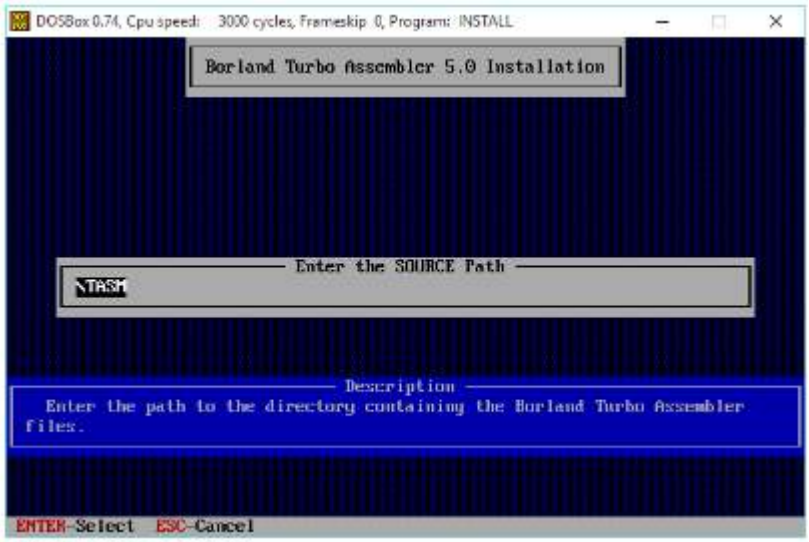
2. Setelah tercopy semua kedalam folder TASM yang dibuat tadi Bukalah aplikasi Dosbox yang telah terinstall tadi.
3. || ketik mount c: c:\
   || lalu ketik mount d: d:\ sampai keluar pesan **"Drive D is mounted as local directory D:\"**



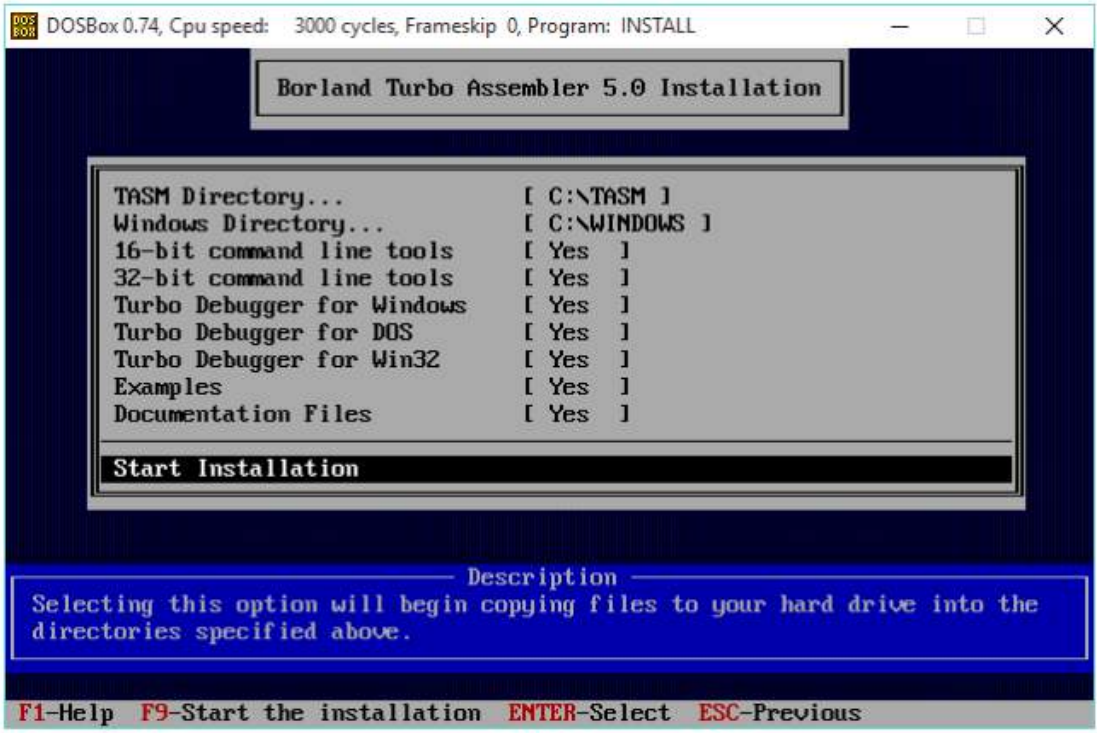
4. Setelah itu pindah ke direktory C: dengan perintah C:
   kemudian masuk ke folder TASM yang di drive C tadi dengan perintah seperti pada gambar di atas
   || cd TASM
5. Kemudian ketik install dan tekan enter. Maka akan keluar seperti di gambar ini :
   C:\TASM>install



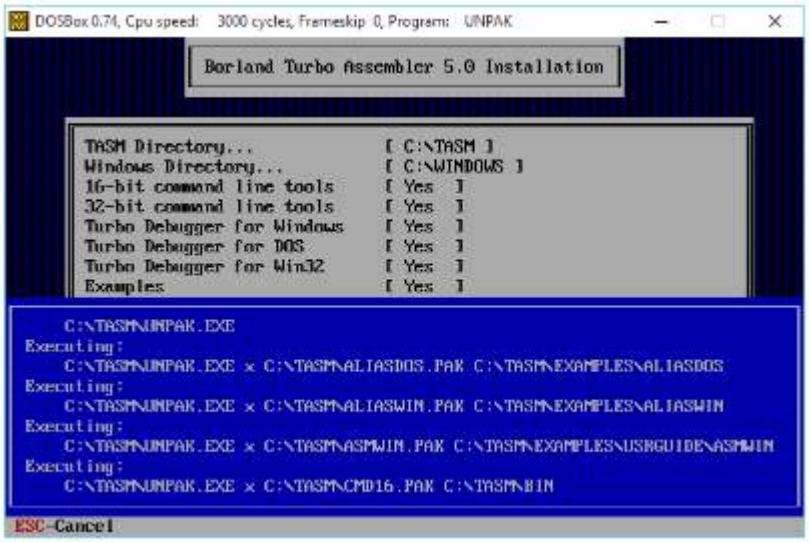
6. Tekan ENTER
7. Kemudian ketik lokasi drive C, karena lokasi file turbo assembler di drive C dan enter, dan akan keluar seperti ini, tekan Enter.



8. Geser ke bawah Start Installation dan enter , seperti pada gambar di bawah ini :

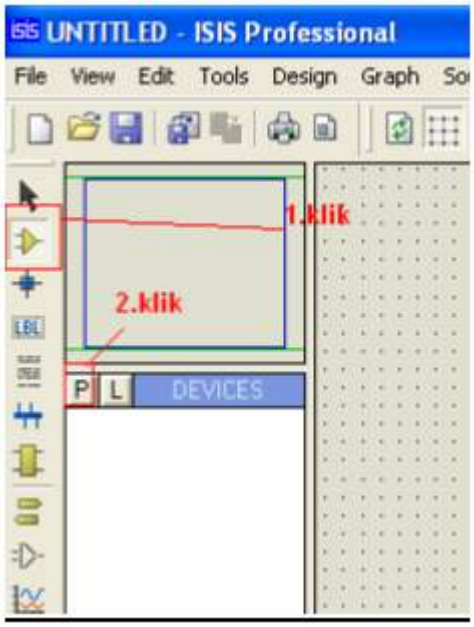


9. Akan keluar seperti gambar dibawah ini dan tunggu sampai selesai :

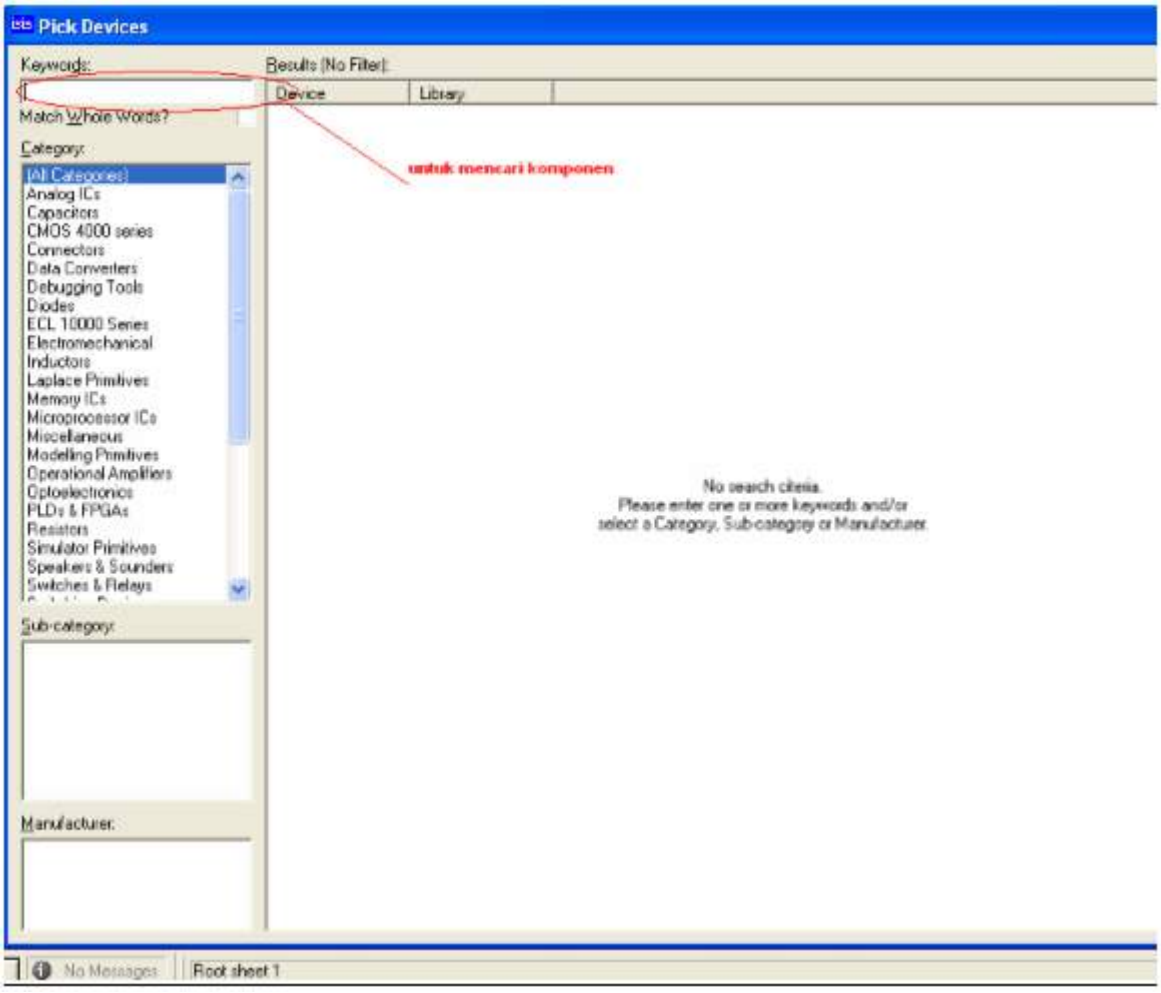


C. Penggunaan Program Proteus.

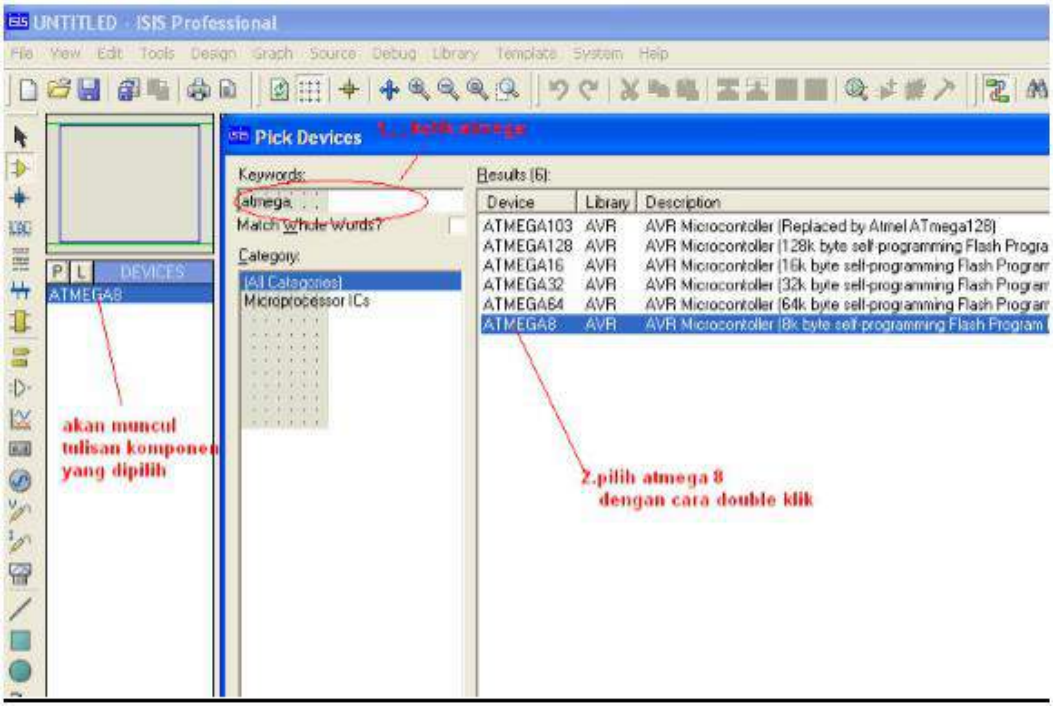
1. Untuk membuka Program Proteus klik All Application, klik Proteus, klis ISIS.
2. Untuk mencari komponen klik Component Mode, klik P seperti pada gambar dibawah.



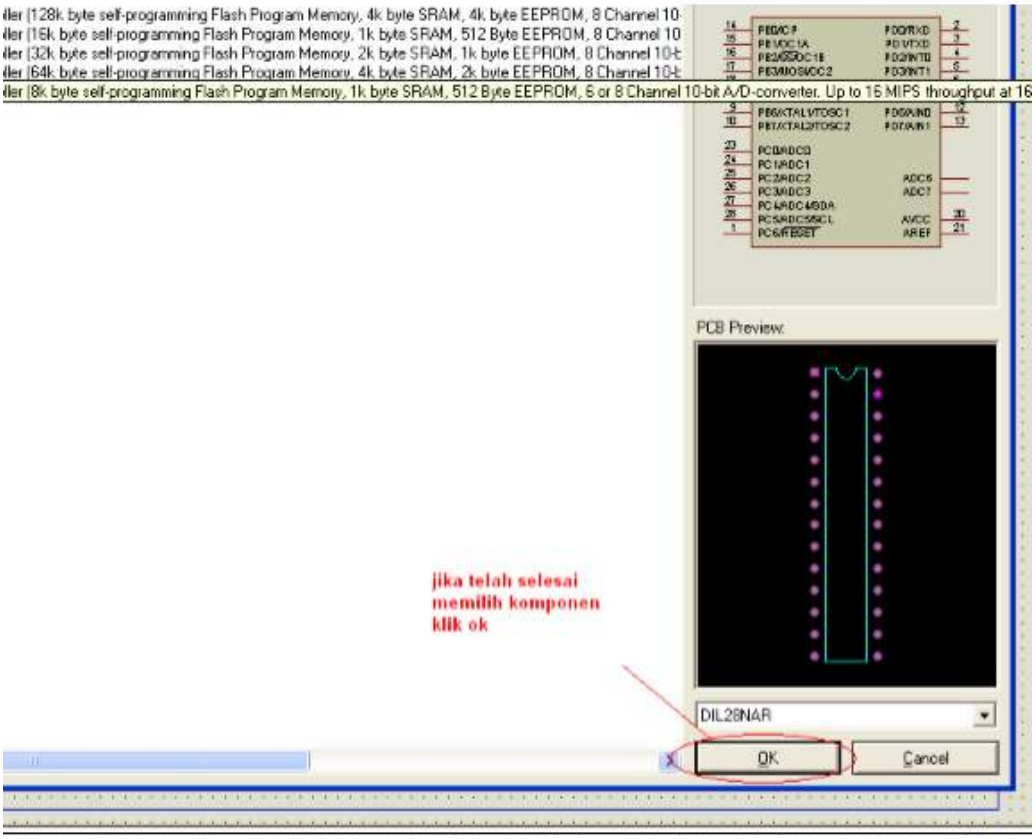
3. Ketikkan komponen yang akan ditampilkan pada kolom dibawah Keywords : seperti gambar dibawah ini .

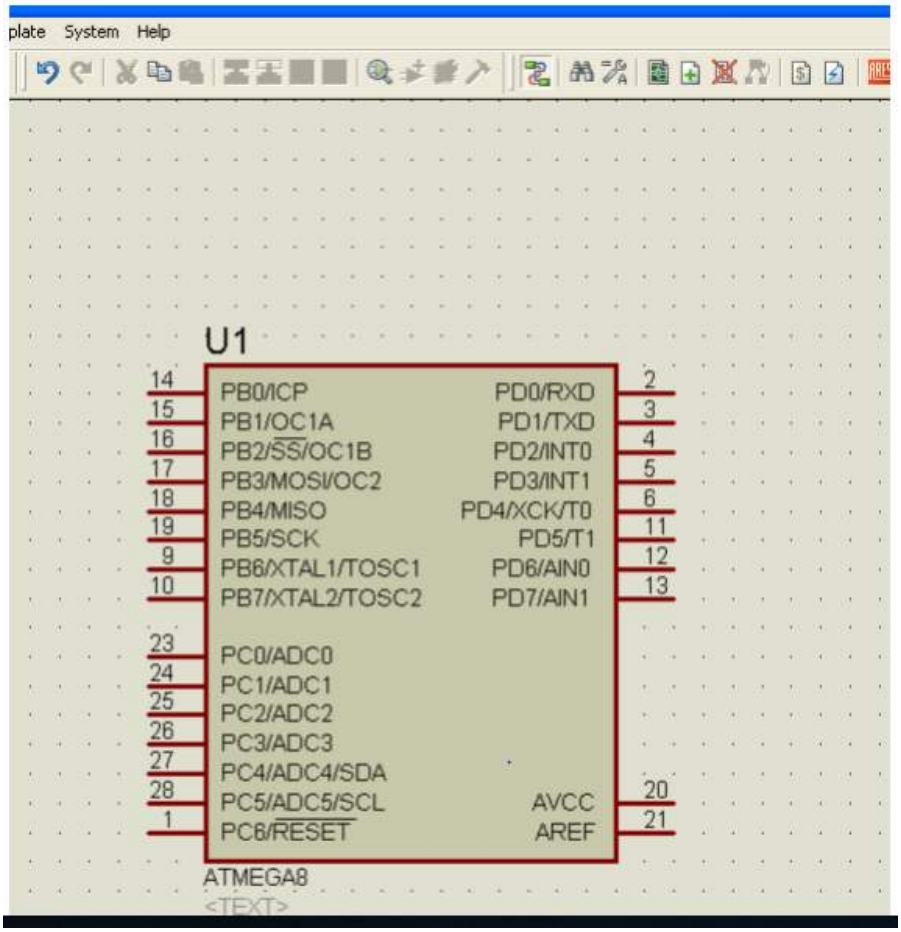


4. Akan muncul result, lalu klik komponen yang dipilih akan muncul komponen di bawah P L

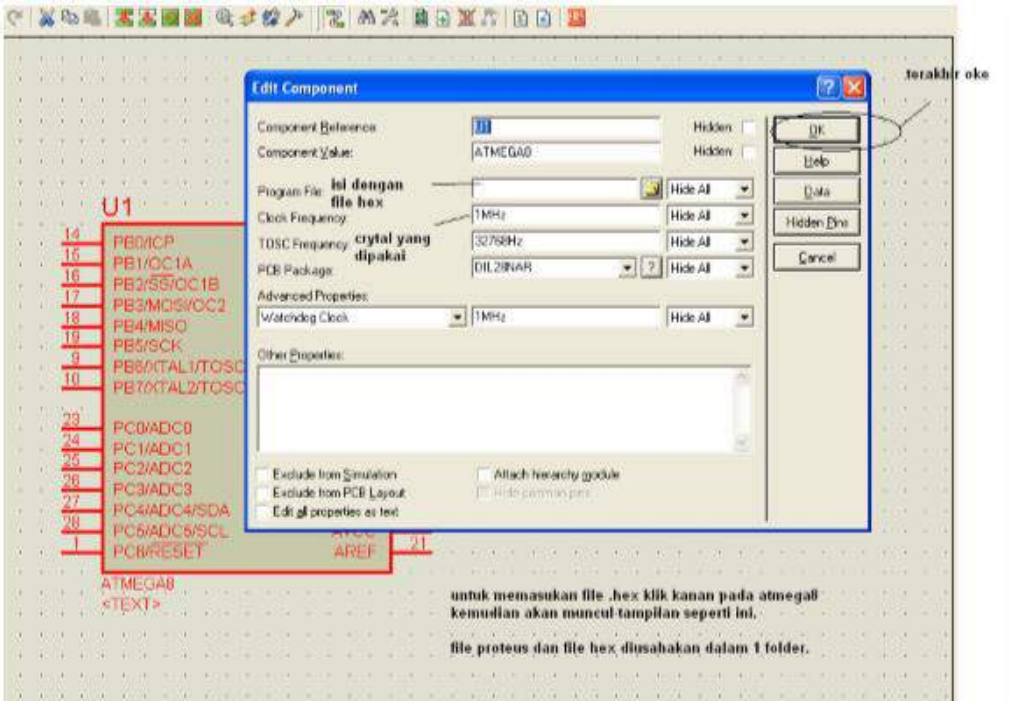


5. Setelah komponen yang dipilih di klik 2 kali, lalu pilih ok

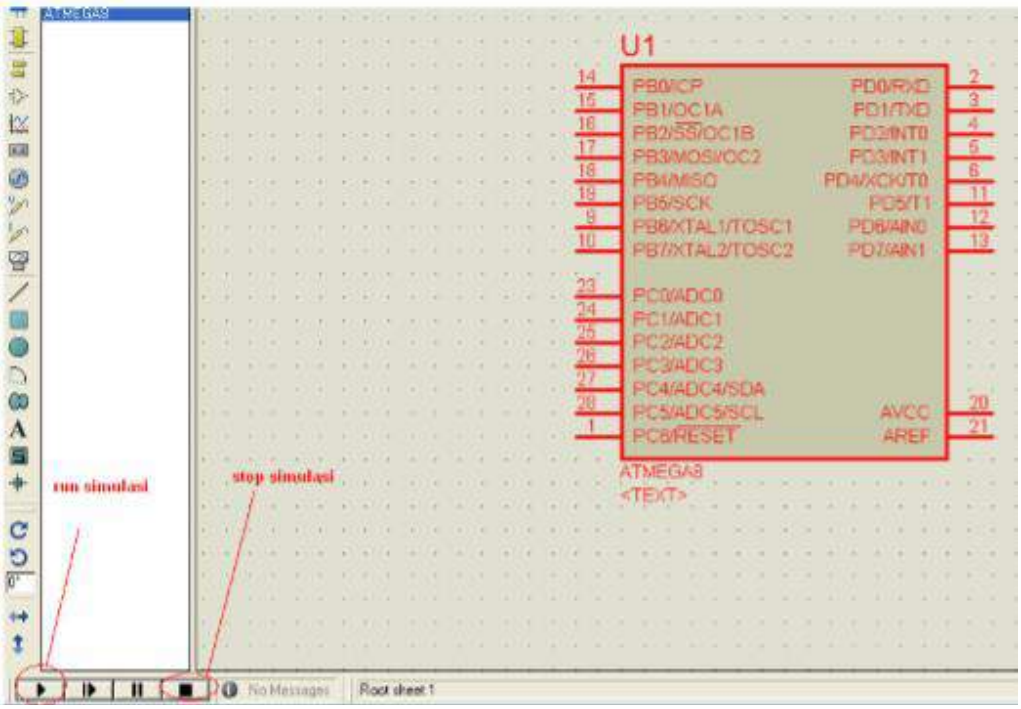




6. Memasukkan file hex ke IC Mikrokontroller hasil kompilasi dari CVAVR klik 2 kali pada IC nya dan browse file tempat meletakkan file hex tersebut, seperti terlihat pada gambar di bawah ini.

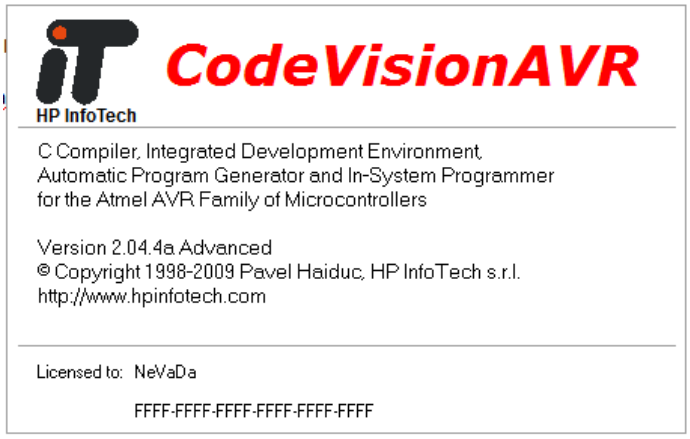


7. Untuk menjalankan simulasi klik icon Run The Simulasi dan untuk menghentikan simulasi klik icon Stop Simulasi tersebut, seperti terlihat pada gambar di bawah ini.

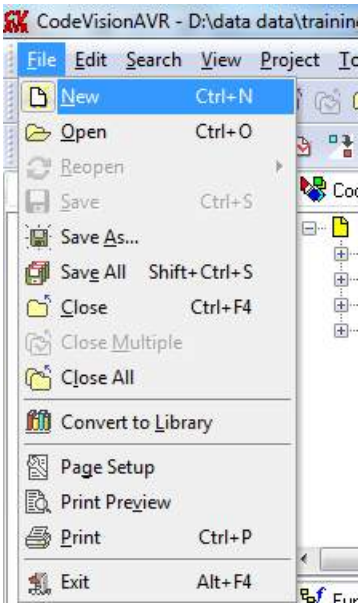


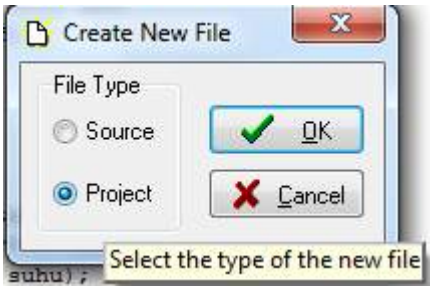
D. Penggunaan Program CVAVR.

1. Jalankan codevision

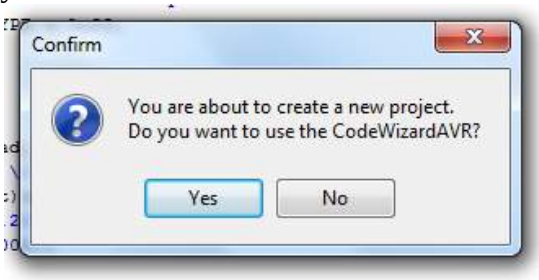


2. Pilih file-new. Kemudian muncul box dialog. Pilih project.

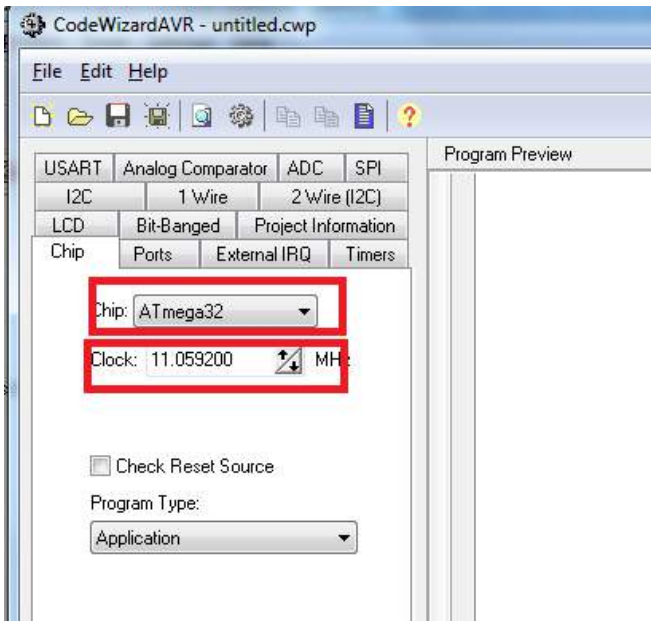




3. Kemudian kita diberi pilhan apakah menggunakn wizar atau tidak. Jika ya pilih yes

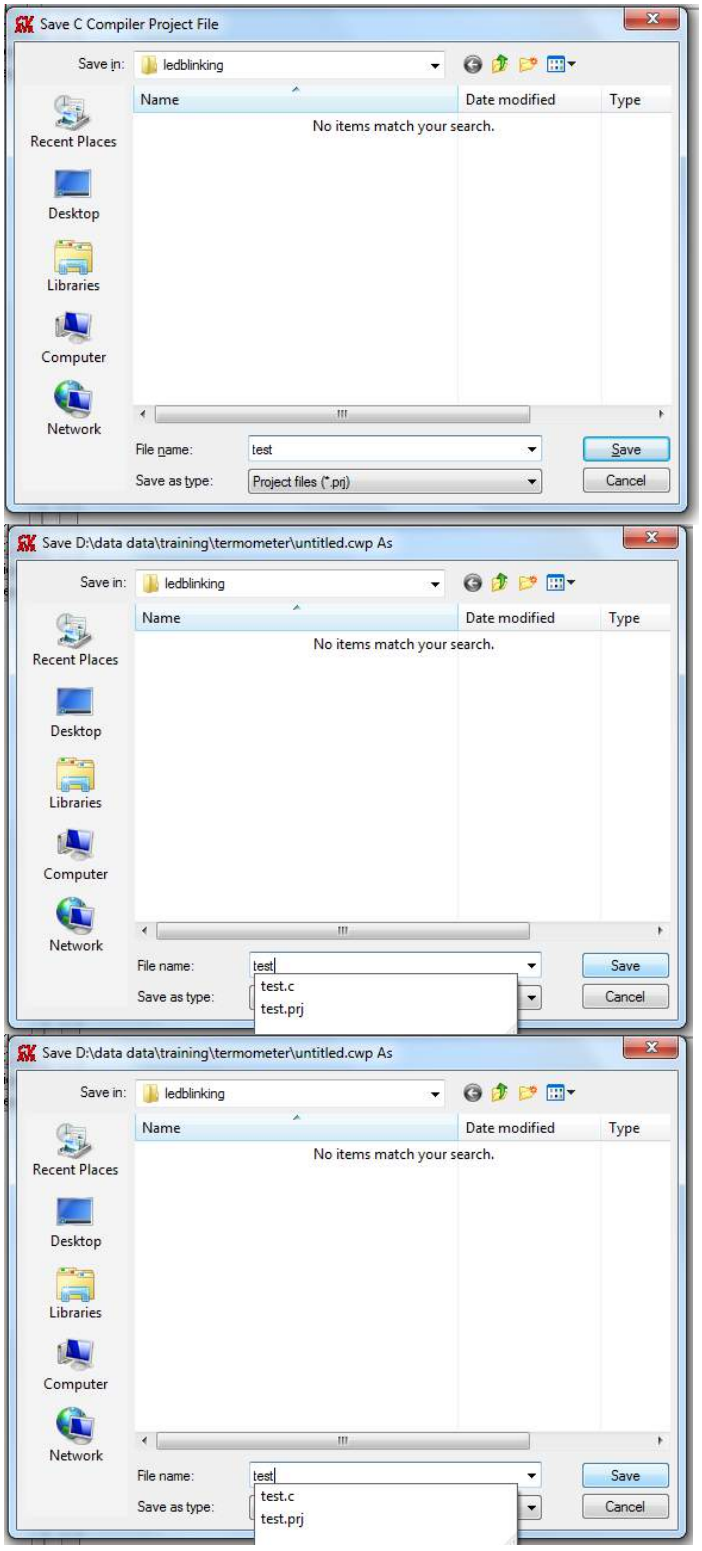


4. Kemudian akan muncul box dialog. Kita harus pilih ic yang akan di pakai. Missal atmega 32.

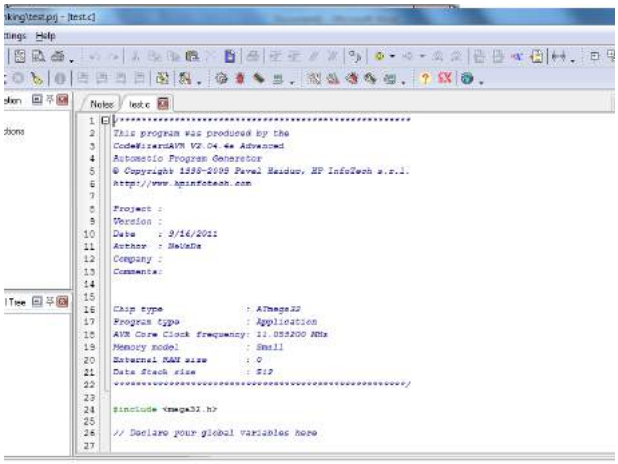


5. Banyak tab-tab yang perlu di setting , sementara kita belum menyeting tab dulu.
6. Pilih file-generate, save and exit.

7. Kemudian proses penyimpanan file dilakukan sebanyak 3 kali. Masing masing dengan eksetensi *.c, *.prj dan *.cwp. usahakan semua nama file sama misal "**test".**



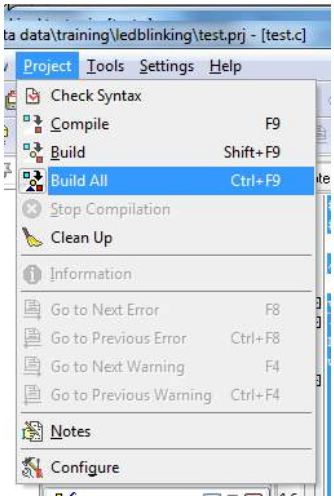
8. Setelah proses penyimpan selesai kita disinilah kita akan menulis program



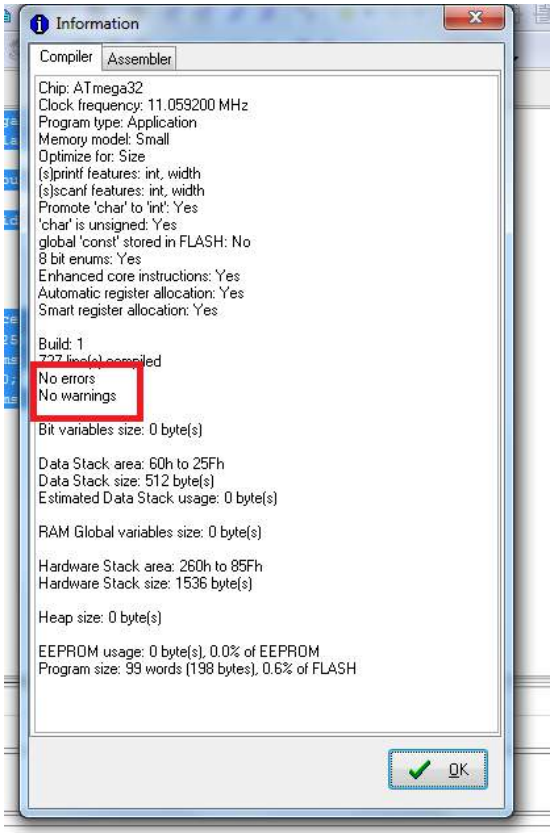
9. Berikut contoh listing program pertama kita

```
#include <mega32.h>
#include <delay.h>
void main(void)
{
DDRB=255;
while (1)
      {
      // Place your code here
      PORTB=255;
      delay_ms(1000);
      PORTB=0;
      delay_ms(1000);
      };
}
```

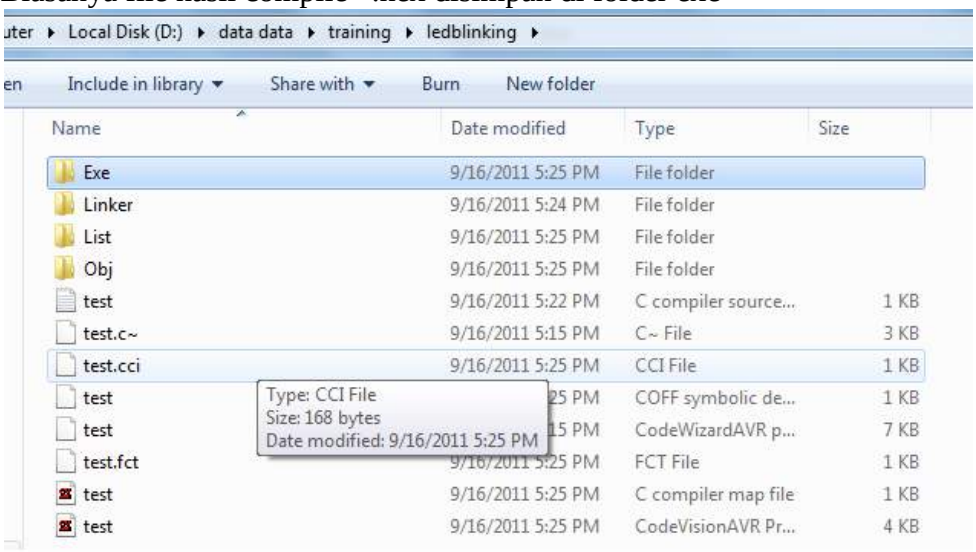
10. Jika sudah selesai kita compile. Hasil dari compile akan menghasilkan file *.hex



11. Berikut jika penulisan syntax program sudah benar



12. Biasanya file hasil compile *.hex disimpan di folder exe



VI. Tugas Akhir

1. Buatlah contoh program sederhana dengan Tasm
2. Buatlah contoh program simulasi dengan Proteus dan CVAVR

VII. Analisis Hasil Percobaan . ……..

VIII. Kesimpulan . ……

MODUL II
PROGRAM COM DAN EXE

I. TUJUAN :

a. Memahami komponen-komponen yang ada di Turbo Assambler.
b. Memahami bentuk-bentuk program yang berextensi .com dan exe.
c. Memahami perintah-perintah perulangan (looping).
d. Memahami perintah-perintah control program.

II. ALAT DAN BAHAN :

a. 1 Unit Personal Computer (PC) / Laptop.
b. Notepad.
c. Turbo Assambler.

III. DASAR TEORI

a. Pengertian Interrupt

Interupsi adalah suatu permintaan khusus kepada mikroposesor untuk melakukan sesuatu. Bila terjadi interupsi, maka komputer akan menghentikan dahulu apa yang sedang dikerjakannya dan melakukan apa yang diminta oleh yang menginterupsi.

b. Vektor Interupsi

Setiap interrupt akan mengeksekusi interrupt handlernya masing-masing berdasarkan nomornya. Sedangkan alamat dari masing- masing interupt handler tercatat di memori dalam bentuk array yang besar elemennya masing-masing 4 byte. Keempat byte ini dibagi lagi yaitu 2 byte pertama berisi kode offset sedangkan 2 byte berikutnya berisi kode segmen dari alamat interupt handler yang bersangkutan. Jadi besarnya array itu adalah 256 elemen dengan ukuran elemen masing-masing 4 byte. Total keseluruhan memori yang dipakai adalah sebesar 1024 byte (256 x 4 = 1024) atau 1 KB dan disimpan dalam lokasi memori absolut 0000h sampai 3FFh. Array sebesar 1 KB ini disebut Interupt Vector Table (Table Vektor Interupsi). Nilai-nilai yang terkandung pada Interupt Vector Table ini tidak akan sama di satu komputer dengan yang lainnya.

Didalam pemrograman dengan bahasa assembler kita akan banyak sekali menggunakan interupsi untuk menyelesaikan suatu tugas.

e. Text Editor

Untuk menuliskan source file untuk program assembly bisa anda gunakan berbagai editor, misalkan Notepad, SideKick, WordStar dan Word Perfect. Source file yang diketikkan harus berupa file ASCII, file ini bisa anda hasilkan melalui WordStar dengan file 'NON DOCUMEN', atau dengan SideKick

f. Compiler

Source file ASCII yang telah anda ketikkan perlu dicompile kebentuk file object dengan extensi .OBJ, dari file object inilah nantinya dapat dijadikan kebentuk file .EXE atau .COM.

Untuk mengcompile source file, misalnya file COBA.ASM menjadi file object dengan extensi .OBJ bisa anda gunakan file TASM.EXE dengan mengetikkan

g. Linkking

File object yang telah terbentuk dengan TASM, belum dapat dieksekusi secara langsung. Untuk membuat file object ke bentuk file yang dapat dieksekusi(ektensi .COM atau .EXE) bisa digunakan file TLINK.EXE.

h. Perbedaan Program COM dan EXE

Program dengan ektensi COM dan EXE mempunya berbagai perbedaan yang menyolok, antara lain :

PROGRAM COM :

- Lebih pendek dari file EXE
- Lebih cepat dibanding file EXE
- Hanya dapat menggunakan 1 segmen
- Ukuran file maksimum 64 KB (ukuran satu segment)
- sulit untuk mengakses data atau procedure yang terletak pada segment yang lain.
-100h byte pertama merupakan PSP(Program Segment Prefix) dari program tersebut.
- Bisa dibuat dengan DEBUG

PROGRAM EXE :

- Lebih panjang dari file COM
- Lebih lambat dibanding file COM
- Bisa menggunakan lebih dari 1 segmen
- Ukuran file tak terbatas sesuai dengan ukuran memory.
- mudah mengakses data atau procedure pada segment yang lain.
- Tidak bisa dibuat dengan DEBUG

IV. TUGAS PENDAHULUAN

1. Apa yang anda ketahui tentang Turbo Assambler
2. Jelaskan dengan singkat macam-macam interrupt
3. Sebutkan DOS interrupt dan BIOS interrupt
4. Jelaskan dengan singkat model program .COM dan model program .EXE

V. PROSEDUR DAN PENGAMATAN PERCOBAAN

1. Program .COM

A. Program menampilkan Bilangan Prima.

Prosedur percobaan :

1. Buka Notepad
2. Ketiklah listing program prima
3. Simpan di folder BIN dengn nama file **prima** dan extensi .asm
   C:\TASM\BIN>prima.asm
4. Buka DOSBox lalu mounting folder C
5. Ketikkan : Z:\> mount c: c:\ <<tekan enter>>
6. Ketikkan : Z:\>**C:** <<tekan enter>>
7. Ketikkan : C:\> **cd TASM** <<tekan enter>>
8. Ketikkan : C:\TASM> **cd BIN** <<tekan enter>>
9. Ketikkan : C:\TASM\BIN>**tasm prima** <<tekan enter>> amati dan catat.
10. Ketikkan : C:\TASM\BIN>**dir prima.*** <<tekan enter>> amati dan catat.
11. Ketikkan : C:\TASM\BIN>**tasm/t prima** <<tekan enter>> amati dan catat.
12. Ulangi **langkah 10**. Amati dan catat hasilnya.

13. Ketikkan : C:\TASM\BIN>**prima** <<tekan enter>> amati dan catat.
14. Ketikkan : C:\TASM\BIN>**exit**

B. Program menampilkan ASCII untuk 7 bit.

Prosedure Percobaan
1. Buka Notepad
2. Ketiklah listing program ascii
3. Ulangi Langkah A.3 sampai dengan A.14 dengan nama file **ASCII**

2. Program .EXE

C. Program mencetakstring.

Prosedur percobaan :
1. Buka Notepad
2. Ketiklah listing program **mencetakstring**
3. Simpan di folder BIN dengn nama file **mencetakstring** dan extensi .asm
   C:\TASM\BIN> **mencetakstring**.asm
4. Buka DOSBox lalu mounting folder C
5. Ketikkan : Z:\> mount c: c:\ <<tekan enter>>
6. Ketikkan : Z:\>**C:** <<tekan enter>>
7. Ketikkan : C:\> **cd TASM** <<tekan enter>>
8. Ketikkan : C:\TASM> **cd BIN** <<tekan enter>>
9. Ketikkan : C:\TASM\BIN>**tasm mencetakstring** <<tekan enter>> amati dan catat.
10. Ketikkan : C:\TASM\BIN>**dir mencetakstring.*** <<tekan enter>> amati dan catat.
11. Ketikkan : C:\TASM\BIN>**tasm mencetakstring** <<tekan enter>> amati dan catat.
12. Ulangi **langkah 10**. Amati dan catat hasilnya.
13. Ketikkan : C:\TASM\BIN> **mencetakstring** <<tekan enter>> amati dan catat.
14. Ketikkan : C:\TASM\BIN>**exit**

D. Program merubah huruf kecil menjadi huruf besar.

Prosedure Percobaan
1. Buka Notepad
2. Ketiklah listing program huruf besar
3. Ulangi Langkah C.3 sampai dengan C.14 dengan nama file **hurufbesar**

1. Listing Program : prima

```
Cetak_Klm MACRO Klm        ; Macro untuk mencetak kalimat
          MOV AH,09
          LEA DX,Klm
          INT 21h
          ENDM

CDesimal  MACRO Angka
```

```
        LOCAL Ulang, Cetak
        MOV AX,Angka        ; AX = angka yang akan dicetak
        MOV BX,10           ; BX = penyebut
        XOR CX,CX           ; CX = 0

Ulang :
        XOR DX,DX     ; Cegah sisa bagi menjadi pembilang !
        DIV BX        ; Bagi angka yang akan dicetak dengan 10
        PUSH DX       ; Simpan sisa bagi dalam stack
        INC CX        ; CX ditambah 1
        CMP AX,0      ; Apakah hasil bagi sudah habis ?
        JNE Ulang     ; Jika belum, ulangi lagi !

Cetak :
        POP DX        ; Ambil 1 angka yang disimpan
        ADD DL,'0'    ; Ubah 1 angka dalam kode ASCII
        MOV AH,02     ;
        INT 21h       ; Cetak angka tersebut
        LOOP Cetak    ; ulangi
        ENDM

;/=============================================\;
; Program : PRIMA.ASM                         ;
; Fungsi  : Mencari dan menampilkan angka     ;
;           prima dari 0 sampai 1000          ;
;\=============================================/;

        .MODEL SMALL
        .CODE
        ORG 100h

TData : JMP Awal
        Batas DW 1000
        Prima DW 0
        I DW 2
        J DW 2
        Spasi DB ' $'
        Header DB 9,9,9,'Bilangan Prima 1 sampai 1000 : ',13,10
        DB 9,9,9,'------------------------',13,10,10,'$'

Awal :
        Cetak_Klm Header
Proses :
        MOV AX,Batas        ; Jika bilangan yang dicek
        CMP AX,I            ; sudah sama dengan Batas
        JE Exit             ; maka selesai
ForI :
```

```
        MOV J,2         ; J untuk dibagi oleh I
        MOV Prima,0     ; Prima = Tidak
ForPrima:
        MOV AX,Prima            ;
        CMP AX,0        ; Apakah prima = Tidak ?
        JNE TambahI     ; jika Prima = Ya, lompat ke TambahI
        MOV AX,I        ;
        CMP AX,J        ; I = J ?
        JNE Tidak       ; Jika tidak sama, lompat ke Tidak

        CDesimal I              ; Cetak angka prima
        Cetak_Klm Spasi         ; Cetak spasi
        MOV Prima,1             ; Prima = Ya
        JMP TambahJ             ; Lompat ke TambahJ

Tidak :
        MOV DX,0        ;
        MOV AX,I        ;
        MOV BX,J        ;
        DIV BX          ; Bagi I dengan J
        CMP DX,0        ; Apakah sisa bagi=0?
        JNE TambahJ     ; Jika tidak sama lompat ke TambahJ
        MOV Prima,1     ; Prima = Ya

TambahJ :
        INC J           ; Tambah J dengan 1
        JMP ForPrima    ; Ulangi, bagi I dengan J

TambahI :
        INC I           ; Tambah I dengan 1
        JMP Proses      ; Ulangi Cek I = prima atau bukan

Exit :
        INT 20h
END     TData
```

2. Listing Program : ascii

```
 ;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~;
 ;   PROGRAM: ABC0.ASM              ;
 ;   FUNGSI : MENCETAK 16 BUAH  ;
 ;            KARAKTER DENGAN   ;
 ;            INT 21h SERVIS 02 ;
 ;================================;
        .MODEL SMALL
        .CODE
        ORG 100h
 Proses :
```

```
        MOV AH,02h   ; Nilai servis
        MOV DL,00h   ; DL=karakter 'A' atau DL=41h
        MOV CX,240   ; Banyaknya pengulangan yang akan
Ulang  :
        INT 21h      ; Cetak karakter !!
        INC DL       ; Tambah DL dengan 1
        LOOP Ulang   ; Lompat ke Ulang

        INT 20h
END    Proses
```

3. Listing Program : mencetakstring

```
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~==;
; Program: kal0.asm                    ;
; Fungsi : Mencetak String             ;
;         dengan Int 21 servis 9       ;
;======================================;
        .MODEL SMALL
        .CODE
        ORG 100h
Tdata : JMP Proses
        Kal0 DB 'PROSES PENCETAKAN STRING ',13,10,'$'
        Kal1 DB 'DIBELAKANG TANDA $ TIDAK BISA DICETAK '
Proses:
        MOV AH,09h           ; Servis ke 9
        MOV DX,OFFSET Kal0   ; Ambil Alamat Offset Kal0
        INT 21h              ; Cetak perkarakter sampai tanda $
        LEA DX,Kal0          ; Ambil Alamat Offset Kal0
        INT 21h              ; Cetak perkarakter sampai tanda $
        LEA DX,Kal0+7        ; Ambil Alamat Offset KAl0+7
        INT 21h              ; Cetak perkarakter sampai tanda $
        LEA DX,KAL1          ; Ambil Offset kal1
        INT 21h              ; Cetak perkarakter sampai ketemu $
        INT 20h              ; Selesai, kembali ke DOS
END Tdata
```

4. Listing Program : merubah huruf kecil ke huruf besar

```
;/================================================\;
;       Program : UPCASE.ASM                     ;
;       Fungsi :  Merubah huruf kecil menjadi    ;
;           huruf besar                          ;
;\================================================/;
        .MODEL SMALL
        .STACK 200h
        .DATA
```

```
Klm DB 'Cinta menyebabkan rindu yang paling sengsara $'

        .CODE
Proses:
        MOV AX,@DATA        ; Inialisasi, supaya
        MOV DS,AX           ; DS menunjuk pada data
        XOR BX,BX
        MOV CX,47
Ulang:
        CMP Klm[BX],'a'     ; Apakah huruf kecil ?
        JB Tidak            ; Tidak, cek berikutnya
        CMP Klm[BX],'z'     ; Apakah huruf kecil ?
        JA Tidak            ; Tidak, cek berikutnya
        SUB Klm[BX],'a'-'A' ; Jadikan huruf besar
        Tidak:
        INC BX              ; Akses karakter berikutnya
        LOOP Ulang          ; Ulangi
        Cetak:
        MOV AH,09           ; Cetak kalimat yang telah
        LEA DX,Klm          ; dirubah menjadi huruf besar
        INT 21h             ; semuanya
Exit:
        MOV AX,4C00h        ; Selesai,
        INT 21h             ; kembali ke DOS
END   Proses
```

VI. Tugas Akhir

1. Sebutkan mnemonic instruksi perulangan dari program diatas
2. Sebutkan mnemonic instruksi control program dari program diatas
3. Buatlah struktur program .com
4. Buatlah struktur program .exe

VII. Analisa hasil percobaan tersebut

VIII. Berilah kesimpulan dari hasil percobaan tersebut.

MODUL III
PENGALAMATAN DAN ARITMATIKA

I. TUJUAN :

a. Memahami perpindahan data baik dalam register maupun dalam alamat .
b. Memahami mnemonic aritmatik.
c. Memahami debugging program.

II. ALAT DAN BAHAN :

a. 1 Unit Personal Computer (PC) / Laptop.
b. Notepad.
c. Turbo Assambler.

III. DASAR TEORI

Dalam bahasa assembly, perpindahan data dapat dilakukan dalam berbagai cara yaitu :

1. Perpindahan tak langsung.
2. Perpindahan data dari register ke memori.
3. Perpindahan data memori ke register.
4. Perpindahan data langsung ke register.
5. Perpindahan data langsung ke memori.

Khusus untuk perpindahan data dari atau ke memori, dapat dilaksanakan dengan beberapa macam pengalaman, yaitu :

1. Pengalamatan tak langsung.
2. Pengalamatan secara langsung.
3. Pengalamatan dengan menggunakan indeks ke register.
4. Pengalamatan dengan menggunakan basis ke register.
5. Pengalamatan dengan menggunakan basis dan indeks register.

Untuk melaksanakan perpindahan data, bahasa assembly memiliki beberapa perintah yang dapat dibedakan sebagai perintah untuk memindahkan data string yang berupa deretan aksara. Untuk perintah perpindahan data tunggal, perintah yang dapat digunakan antara lain :

1. MOV, Dipakai untuk memindahkan data dari memori/register ke memori/register atau data langsung ke register.

2. IN/OUT, Untuk memindahkan data dari /ke port. Data yang akan dipindahkan, diberikan pada register AX/AL dan alamat port diberikan pada register DX.

3. PUSH/POP, Dipakai untuk memindahkan data dari/ke stack memori.

4. XCHG, Dipakai untuk menukar data.

5. XLAT, Dipakai untuk mencari data dalam suatu table. Alamat table ada register BX dan sebagai indeks digunakan register AL.

6. PUSHF/POPF, Dipakai untuk memindahkan isi Flag ke/dari stack memori.

7. LEA/LDS/LES,
   LEA untuk memindahkan OFFSET ke register 16 bit.

LDS untuk memindahkan alamat memori ke suatu register dan DS
LES untuk memindahkan alamat memori ke suatu register dan ES.

8. LAHF/SAHF, Untuk memindahkan isi flag register dari/ke register AH. Flag tersebut adalah S, Z, A, P, dan C.

Untuk memindahkan data string, sering dipergunakan perintah-perintah antara lain :
1. MOVS, Untuk memindahkan data string dari DS:SI ke ES:DI.
2. STOS, Untuk memindahkan data string dari AX/AT, ke ES:DI.
3. LODS, Untuk memindahkan data string dari DS:SI ke Akumulator.

Untuk operasi aritmatika dan logika digunakan perintah-perintah sebagai berikut :

1. Perintah ADD/ADC (Penjumlahan)
   Digunakan untuk menjumlahkan 2 operand tanpa menggunakan carry (ADD) atau menggunakan carry (ADC).

2. Perintah SUB/SBB (Pengurangan)
   Digunakan untuk mengurangkan 2 operand tanpa menggunakan carry (SUB) atau menggunakan borrow (SBB).

3. Perintah MUL/IMUL (Perkalian)
   Digunakan untuk mengkalikan data pada akumulator dengan suatu operand dan hasilnya diletakkan pada AX untuk operasi 8 bit atau AX dan BX untuk operasi 16 bit. IMUL dipakai khusus untuk integer bertanda.

4. Perintah DIV/IDIV (Pembagian)
   Digunakan untuk membagikan data pada akumulator dengan suatu operasi dan hasilnya diletakkan pada AX untuk operasi 8 bit atau AX dan BX untuk operasi 16 bit. IDIV dipakai khusus untuk integer bertanda.

5. Perintah INC/DEC (Penaikan/Penurunan data).
   Digunakan untuk menaikkan atau menurunkan data dari suatu register.

6. Perintah AND/OR/NOT/XOR (Logika).
   Digunakan untuk melakukan operasi-operasi logika AND, OR, NOT dan XOR.

7. Perintah SHR/SHL (Pergeseran)
   Dipakai untuk menggeserkan data dari suatu register ke kanan/kiri sebanyak isi register CX.

8. Perintah ROR/ROL/RCL/RCR (Perputaran).
   Dipakai untuk memutar isi register ke kanan/kiri sebesar 1 bit. Jumlah perputaran ditentukan oleh isi register CX, RCL dan RCR merupakan perputaran yang melintasi carry.
   Selain perintah-perintah yang telah disebutkan diatas, masih terdapat beberapa perintah yang umumnya digunakan bersama-sama dengan perintah perpindahan aritmatika.
   Perintah ini misalnya, perintah untuk perbandingan string (CMPS), perintah pencarian string (SCAS), perintah perulangan REPZ, REPE, REPNE, REPNZ. Apabila dalam

memindahkan string memakai operasi byte, maka perintah pemindahan data harus ditambah dengan aksara B seperti MOVSB, sedangkan bila melakukan operasi word, maka dibelakang perlu ditambah dengan aksara W.
Perintah yang menyangkut operasi aritmatika dan logika yang biasanya sering dijumpai adalah perintah CLC (Clear Carry Flag), CMC (Complement Carry), dan STD (Set Carry Flag) yaitu yang berhubungan dengan status flag.

IV. TUGAS PENDAHULUAN

1. Buatlah program dalam bahasa assembly yang dapat mengisi register-register 8 bit :

AH=00	BL=02	CL=04	DL=06
AH=01	BH=03	CH=05	DH=07

2. Ada 4 buah bilangan yang telah ditempatkan pada masing-masing menurut memori seperti berikut :
   - Bilangan -1 sebesar AC52 pada lokasi memori 400 dan 401.
   - Bilangan -2 sebesar E894 pada lokasi memori 402 dan 403.
   - Bilangan -3 sebesar B2F5 pada lokasi memori 406 dan 407.
   - Bilangan -4 sebesar 1B58 pada lokasi memori 40E dan 40F.

   a. Buatlah program **penjumlahan** bilangan -1 dengan bilangan -2 dan hasilnya ditempatkan pada lokasi memori 404 dan 405.
   b. Program tersebut dilanjutkan dengan **pengurangan** bilangan -1 dengan bilangan -3 dan hasilnya diletakkan pada lokasi 408 dan 409.
   c. Selanjutnya diteruskan dengan **perkalian** bilangan -1 dengan bilangan -2 dan hasilnya diletakkan pada lokasi memori 40A sampai dengan 40D.
   d. Kemudian dilanjutkan dengan **pembagian** bilangan hasil **perkalian** tersebut dengan bilangan -4 yang hasilnya diletakkan pada lokasi memori 410 sampai dengan 413.
   e. Siapkan juga program untuk operasi **logika** AND, OR, NOT dan XOR dengan data dari bilangan -1 dan bilangan -2.

V. PROSEDUR DAN PENGAMATAN PERCOBAAN

1. Penjumlahan dan Debug

Prosedur percobaan :

1. Buka Notepad
2. Ketiklah listing program penjumlahan berikut :

```
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~;
;   PROGRAM : TAMBAH.ASM                  ;
;   FUNGSI  : MELIHAT PENAMBAHAN          ;
;           YANG DILAKUKAN                ;
;           OLEH BERBAGAI                 ;
;           PERINTAH                      ;
;=======================Nardi=======      ;

        .MODEL SMALL
        .CODE
        ORG 100h
 Proses :
        MOV AH,15h          ; AH:=15h
        MOV AL,4            ; AL:=4
        ADD AH,AL           ; AH:=AH+AL, jadi AH=19h
```

```
        MOV AX,1234h        ; NIlai AX:=1234h dan carry=0
        MOV BX,0F221h       ; Nilai BX:=F221h dan carry=0
        ADD AX,BX           ; AX:=AX+BX, jadi nilai AX=0455h

        MOV AX,1234h        ; AX = 1234h  CF = 0
        MOV BX,9ABCh        ; BX = 9ABCh  CF = 0
        MOV CX,5678h        ; BX = 5678h  CF = 0
        MOV DX,0DEF0h       ; DX = DEF0h  CF = 0
        ADD CX,DX           ; CX = 3568h  CF = 1
        ADC AX,BX           ; AX = AX+BX+CF = ACF1

        INC AL              ; AL:=AL+1, nilai pada AL ditambah 1

        INT  20h
END     Proses
```

3. Simpan di folder BIN dengn nama file **tambah** dan extensi .asm
   C:\TASM\BIN> **tambah**.asm
4. Buka DOSBox lalu mounting folder C
5. Ketikkan : Z:\> mount c: c:\ <<tekan enter>>
6. Ketikkan : Z:\>**C:** <<tekan enter>>
7. Ketikkan : C:\> **cd TASM** <<tekan enter>>
8. Ketikkan : C:\TASM> **cd BIN** <<tekan enter>>
9. Ketikkan : C:\TASM\BIN>**tasm tambah** <<tekan enter>> amati dan catat.
10. Ketikkan : C:\TASM\BIN>**tasm/t tambah** <<tekan enter>> amati dan catat.
11. Ketikkan : C:\TASM\BIN>**debug tambah.com** <<tekan enter>> amati dan catat.
14. Untuk menampilkan semua isi register dan alamat ketikkan : **-r** <<tekan enter>> amati dan catat.
15. Tris dengan mengetikkan: **-t** <<tekan enter>> amati dan catat perubahannya.
16. Ulang nomor 15 sebanyak 11 kali dan setiap kali tris amati dan catat perubahannya.
17. Ketikkan: q untuk keluar program.

2. Pengurangan

Prosedur percobaan :

1. Buka Notepad
2. Ketiklah listing program pengurangan berikut :

```
;======================================= ;
;  PROGRAM      : KURANG.ASM             ;
;  AUTHOR       : Nardi                  ;
;  FUNGSI       : MENGURANGKAN ANGKA     ;
;                  122EFFF-0FEFFFF       ;
;                                        ;
;======================================= ;

        .MODEL SMALL
        .CODE
        ORG 100h
```

```
TData :
        JMP   Proses        ; Lompat ke Proses
        ALo   EQU 0EFFFh
        AHi   EQU 122h
        BLo   EQU 0FFFFh
        Bhi   EQU 0FEh
        HslLo DW  ?
        HslHi DW  ?
Proses :
        MOV  AX,ALo         ; AX=EFFFh
        SUB  AX,Blo         ; Kurangkan EFFF-FFFF, jadi AX=F000
        MOV  HslLO,AX       ; HslLo bernilai F000
        MOV  AX,AHi         ; AX=122h
        SBB  AX,BHi         ; Kurangkan 122-FE-Carry, AX=0023
        MOV  HslHi,AX       ; HslHi bernilai 0023

        INT 20h             ; Kembali ke DOS
END     TData
```

3. Simpan di folder BIN dengn nama file **kurang** dan extensi .asm
   C:\TASM\BIN> **kurang**.asm
4. Jadikanlah program com dengan tasm dan tlink/t.
5. Untuk melihat kebenarannya dapat digunakan debug.
6. Ketikkan \\debug kurang.com
7. Ketikkan -r amati dan catat
8. Tris dengan mengetikkan –t sampai muncul <<program terminate>> amati dan catat
9. Ketikkan –**q** untuk berhenti program.

3. Perkalian

Prosedur percobaan :

1. Buka Notepad
2. Ketiklah listing program perkalian berikut :

```
;================================== ;
;  PROGRAM : KALI.ASM                ;
;  AUTHOR    : Nard                  ;
;  FUNGSI    : MENGALIKAN BILANGAN   ;
;              16 BIT, HASIL         ;
;              PADA DX:AX            ;
;================================== ;

     .MODEL SMALL
     .CODE
     ORG 100h
TData :
      JMP   Proses        ; Lompat ke Proses
      A     DW  01EFh
      B     DW  02FEh
      HslLo DW  ?
      HslHi DW  ?
```

```
Proses:
    MOV  AX,A           ; AX=1EF
    MUL  B              ; Kalikan 1FH*2FE
    MOV  HslLo,AX       ; AX bernilai C922 sehingga HslLo=C922
    MOV  HslHi,DX       ; DX bernilai 0005 sehingga HslHi=0005

    INT 20h             ; Kembali ke DOS
END   TData
```

3. Simpan di folder BIN dengn nama file **kali** dan extensi .asm
   C:\TASM\BIN> **kali**.asm
4. Jadikanlah program com dengan tasm dan tlink/t.
5. Untuk melihat kebenarannya dapat digunakan debug.
6. Ketikkan \\debug **kali**.com
7. Ketikkan -r amati dan catat
8. Tris dengan mengetikkan –t sampai muncul <<program terminate>> amati dan catat
9. Ketikkan –**q** untuk berhenti program.

4. Pembagian

Prosedur percobaan :

1. Buka Notepad
2. Ketiklah listing program pembagian berikut :

```
;============================;
;  PROGRAM : BAGI.ASM        ;
;  AUTHOR  : Nardi           ;
;  FUNGSI  : MEMBAGI BILANGAN ;
;            16 BIT, HASIL   ;
;            PADA DX:AX      ;
;============================;

     .MODEL SMALL
     .CODE
     ORG 100h
TData :
    JMP   Proses       ; Lompat ke Proses
    A     DW  01EFh
    B     DW  2
    Hsl   DW  ?
    Sisa  DW  ?
Proses:
    SUB  DX,DX          ; Jadikan DX=0
    MOV  AX,A           ; AX=1EF
    DIV  B              ; Bagi 1EF:2
    MOV  Hsl,AX         ; AX bernilai 00F7 sehingga Hsl=00F7
    MOV  Sisa,DX        ; DX berisi 0001 sehingga Sisa=0001

    INT 20h             ; Kembali ke DOS
END   Tdata
```

3. Simpan di folder BIN dengn nama file **bagi** dan extensi .asm
   C:\TASM\BIN> **bagi**.asm
4. Jadikanlah program com dengan tasm dan tlink/t.
5. Untuk melihat kebenarannya dapat digunakan debug.
6. Ketikkan \\debug **bagi**.com
7. Ketikkan -r amati dan catat
8. Tris dengan mengetikkan –t sampai muncul <<program terminate>> amati dan catat
9. Ketikkan –**q** untuk berhenti program

VI. Tugas Akhir

1. Bagaimanakah bila perintah ADD, SUB, MUL, DIV diganti dengan ADC, SBB, IMUL, dan IDIV ? Apakah perbedaannya ? Mengapa demikian ?

2. Masukkan program pada Tugas Pendahuluan nomor satu (no.1) poin b yang telah anda buat, dan sambungkanlah dengan penulisan program berikut :

```
MOV     AX,[400]
XCHG    AH,AL
MOV     [300],AH
STOSB
INC     DI
STOSW
PUSH    BX
MOV     BX,[BX+SI]
POP     DX
```

3. Jalankan program lengkap tersebut dengan perintah T, amatilah dan catatlah perubahan yang terjadi pada setiap register dan memori yang bersangkutan.

VII. Analisa hasil percobaan tersebut

VIII. Berilah kesimpulan dari hasil percobaan tersebut.

MODUL IV
DOWNLOADER

I. TUJUAN :

1. Dapat membuat downloader sendiri.
2. Memahami system minimum berbasis mikrokontroller.
3. Memahami file-file yang dibutuhkan dalam mengisi dan mengektrak IC mikrokontroller.

II. ALAT DAN BAHAN :

1. Capasitor : 4.7uf/16V, 100nf, 22pf, 22pf
2. Resistor : 2K2, 68R, 68R, 1K, 1K, 10K
3. Dioda : 3.6V, 3.6V
4. Led : 1 merah, 1 hijau
5. Kristal : 12MHZ
6. Atmega 8 + soket
7. Saklar
8. Proteus Desain PCB / Eagle
9. AVR DUDE
10. BASCOM-AVR
11. Khazama AVR Programer
12. eXtreme Burner - AVR
13. Board USBasp
14. Sistem minimum atmega8/48
15. Downloader USBasp/ISP yang mendukung . MISO, MOSI, SCK, Reset AVR
16. Komputer / PC

III. DASAR TEORI

USBasp adalah sebuah downloader untuk mikrokontroler AVR, yang tersusun menggunakan sebuah IC ATMega48 atau ATMega8 dan beberapa komponen pasif. Fungsinya adalah untuk menjembatani atau untuk mengisi program (file hex) hasil compile dari komputer ke Mikrokontroller target.

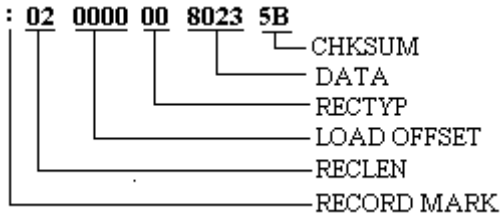
Format Hex dari Intel

Kode-kode program yang diisikan ke Flash PEROM adalah kode-kode biner hasil kompilasi dari program assembler. Ada beberapa format hasil kompilasi dari program assembler, sedangkan format yang paling banyak dipakai adalah format HEX dari Intel.

Anatomi baris–baris dalam file format HEX dari Intel adalah seperti terlihat pada table 1 dan gambar 1 berikut ini.

Table 1. Anatomi baris-baris dalam file format HEX

RECORD MARK	RECLEN	LOAD OFFSET	RECTYP	DATA	CHKSUM
1-byte	1-byte	2-bytes	1-byte	n-bytes	1-byte



Gambar 1. Contoh baris dalam file format HEX

Rincian dari format HEX tersebut adalah sebagai berikut :

1. Huruf pertama dalam baris file format HEX selalu berisi RECORD MARK yang berisi 03AH yang merupakan kode ASCII untuk tanda ":". Tanda ini merupakan tanda identitas file HEX.
2. Huruf Selanjutnya (huruf ke-2 dan ke-3) adalah RECLEN yang dipakai untuk menyatakan banyaknya data dalam baris. Satu byte data dinyatakan dengan dua karakter ASCII sehingga banyaknya data dalam 1 baris maksimal adalah FF (atau decimal 255).
3. Huruf selanjutnya adalah 2 byte untuk LOAD OFFSET yang dipakai untuk menyatakan alamat awal tempat penyimpanan kode-kode dalam baris tersebut.
4. Huruf selanjutnya 1 byte untuk RECTYP yang dipakai untuk menyatakan jenis data. Nilai 00 dipakai untuk menyatakan baris tersebut yang berisikan data biasa. 01 menyatakan bahwa baris tersebut merupakan baris terakhir.
5. Huruf selanjutnya adalah DATA yang dapat berisi nol atau beberapa byte yang dikodekan sebagai pasangan huruf heksa-desimal. Jumlah byte data yang dikodekan sebagai pasangan huruf heksa-desimal adalah sesuai dengan isi dari RECLEN.
6. Baris terakhir merupakan CHKSUM yang dipakai untuk check-sum. Byte-byte RECLEN, LOAD OFFSET. RECTYP, dan byte-byte DATA diatas dijumlahkan. Hasil penjumlahan dibalik (inverted) sebagai bilangan check sum. Hasil penjumlahan bias menghasilkan nilai yang lebih besar dari 2 bilangan heksadesimal, namun hanya 2 bilangan heksadesimal yang bobotnya terkecil yang dipakai.

Pada contoh baris file format HEX di atas dapat diketahui bahwa jumlah data adalah 2 byte yang dapat diketahui dari isi RECLEN, yaitu 02 heksa. Alamat awal data tersebut ditempatkan pada 0000 heksa yang dapat dilihat dari isi RECTYP, yaitu 00. Data dalam baris ini adalah 80 heksa untuk byte pertama dan 23 heksa untuk byte kedua. Check-sum dari baris ini adalah 5B heksa.

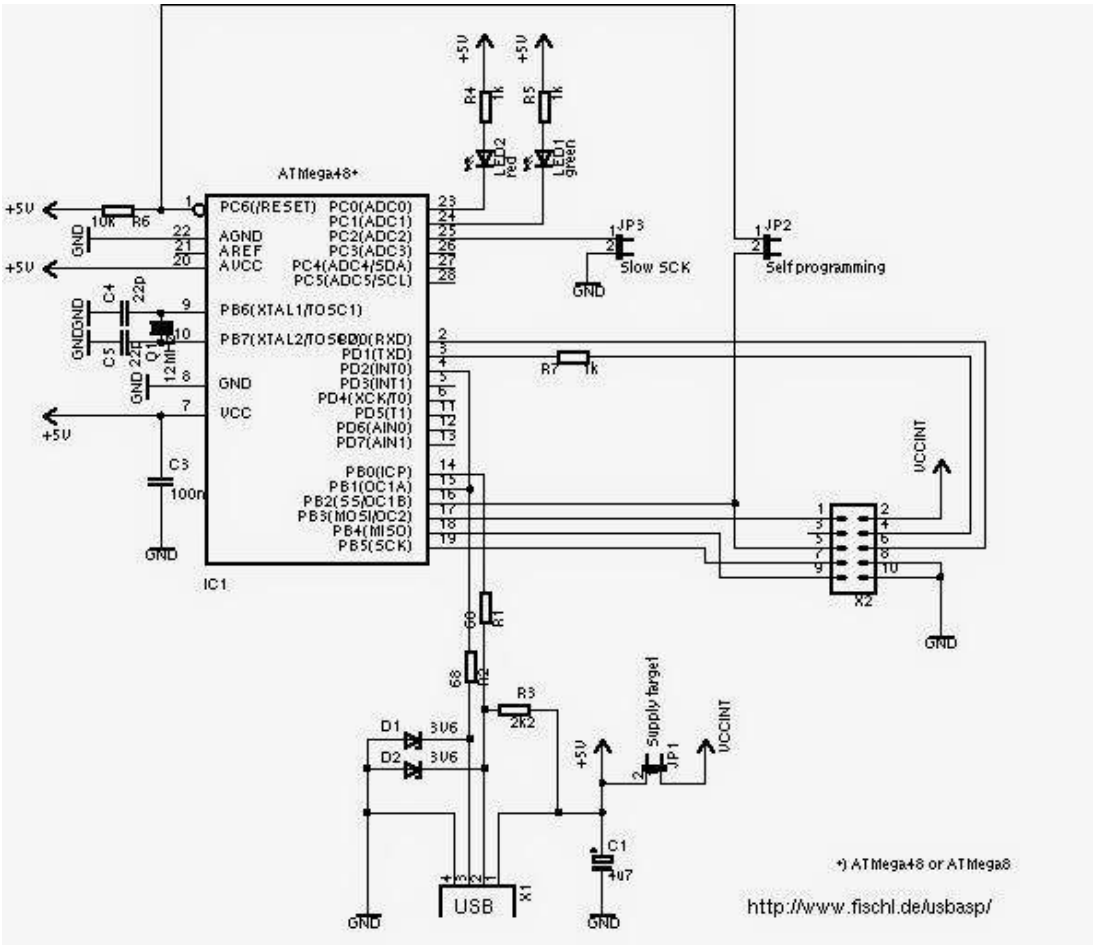
IV. TUGAS PENDAHULUAN

1. Buatlah rangkaian IO system mikroprosesor
2. Jelaskan Anatomi baris-baris format file .hex

V. PROSEDUR DAN PENGAMATAN PERCOBAAN

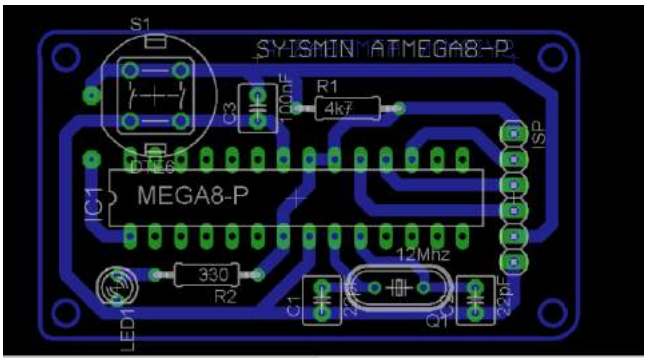
Prosedur percobaan :

1. Buatlah rangkaian seperti gambar 2, untuk contoh skema rangkaian lain dapat mengunjungi http://www.fischl.de/usbasp/



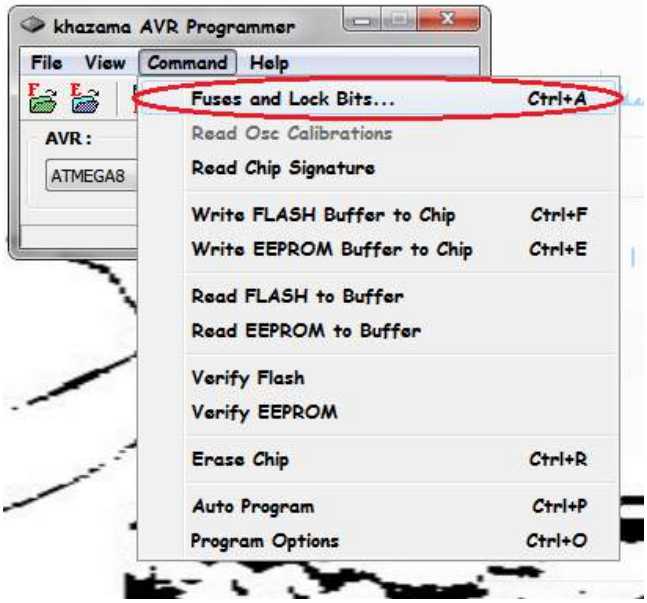
Gambar 2. Skema rangkaian downloader

2. Buat board USBasp seperti gambar 3, menggunakan software pcb eagle.

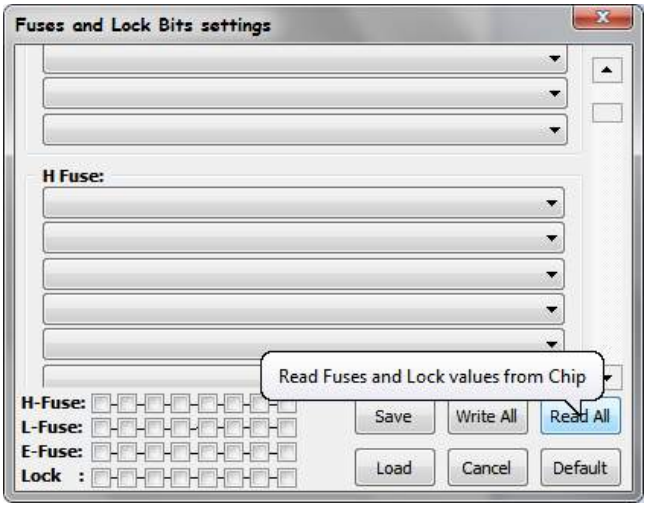


Gambar 3. board USBasp

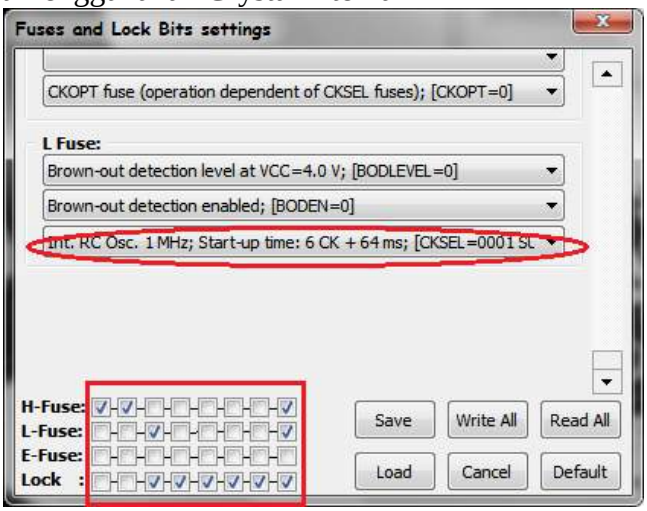
3. Solder komponen-komponennya ke dalam board USBasp.
4. Koneksikan Board USBasp dengan Downloader. Pastikan terkoneksi dengan baik.
5. Buka software downloader (*khazama/avrdude/Extreme burner AVR dll*)
6. Atur fusebitnya seperti berikut:
   a. Fuse and Lock Bits



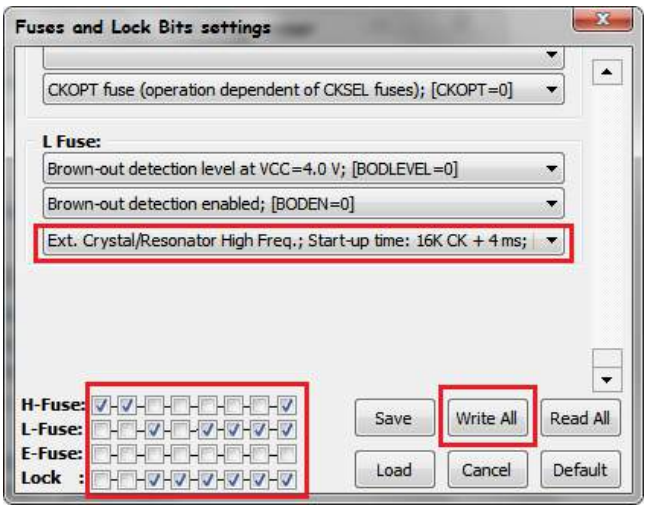
b. Read All



c. Setingan default menggunakan Crystal internal



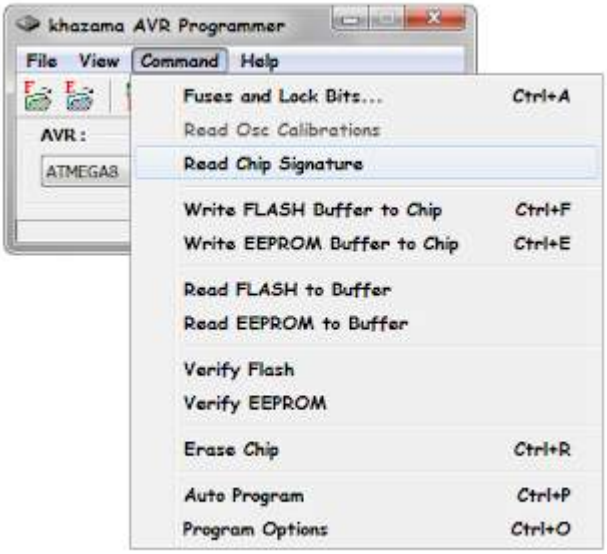
d. Setingan dengan Crystal external



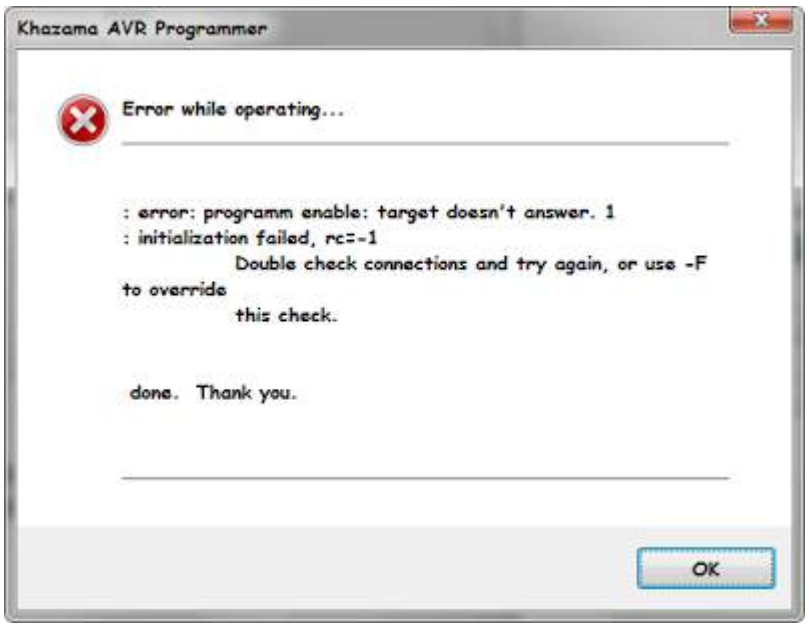
e. Bila sudah selesai diseting klik write all

7. Mengecek apakah downloadernya sudah terkoneksi.

a. Buka Program **Khazama** pilih **command > read signature**



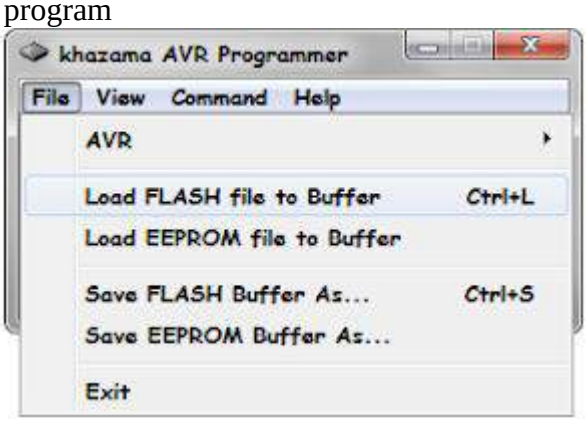
b. Bila ada permasalahan koneksi maka akan muncul notifikasi berikut:



c. Cek kabel koneksi downloader dengan mikro target, dan tegangan supply pada mikro target,
d. Bila masih ada masalah cek kaki2 ic mikro apakah sudah tersambung dengan baik.
e. Cek lagi koneksinya, bila masih ada masalah juga instal drivernya lagi

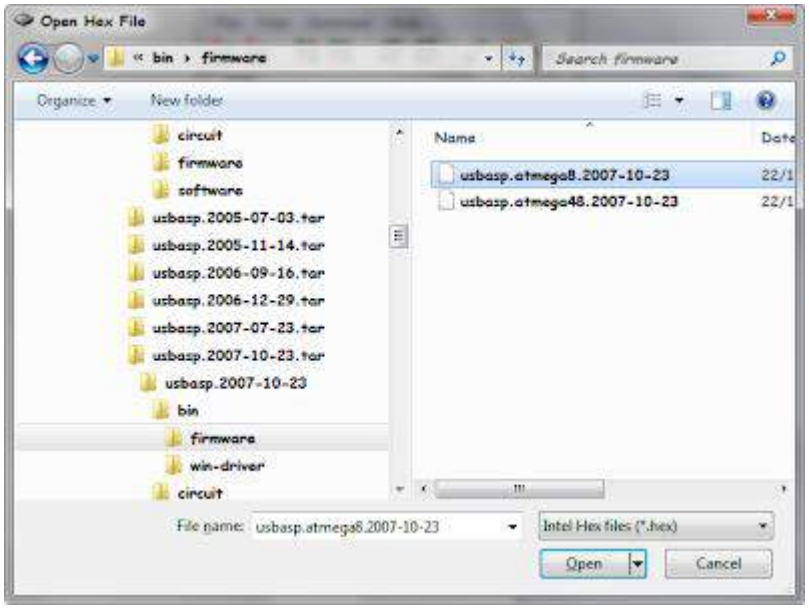
8. Bila koneksi sudah benar akan muncul notifikasi sebagai berikut :

9. Isi/load frimware USBasp ke mikrokontroler target (atmega8)
klik File > Load flash to buffer > kemudian cari dimana firmware tersebut disimpan > setelah itu klik auto program

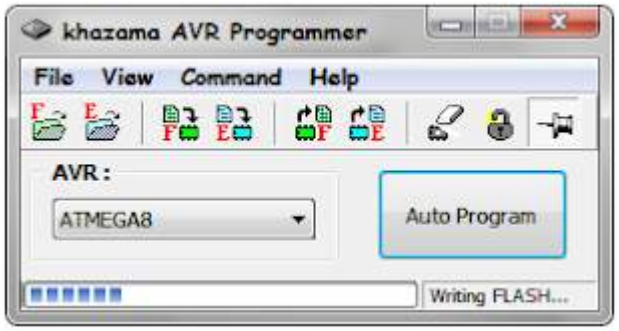


10. Cari file(.hex) firmware.

11. Klik Open

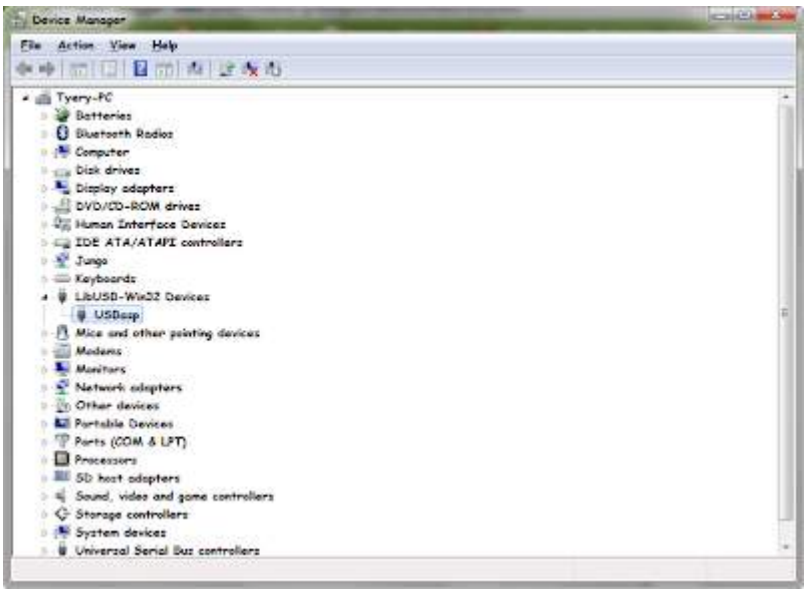


12. Klik Auto Program, Tunggu sampai writing Flash selesai.



13. Bila writing flash sukses maka akan muncul notifikasi berikut :

14. Lapaskan IC atmega8nya dan masukan pada board USBasp yang sudah dibuat, untuk mengetahui apakah usb sudah terkoneksi dengan benar cek di **computer** > **properti** > **device manager**



VI. Tugas Akhir

1. Sebutkan proses-proses dalam praktikum downloader ini.
2. Apa yang anda ketahui tentang system clock
3. Apa yang anda ketahui tentang Flash PEROM.
4. Apa yang anda ketahui tentang Kristal internal dan Kristal external

VII. Analisa hasil percobaan tersebut

VIII. Berilah kesimpulan dari hasil percobaan tersebut.

MODUL V
INPUT OUTPUT DIGITAL

I. TUJUAN :

1. Mengakses perangkat digital I/O
2. Mengerti struktur pemrograman C di CodeVisionAVR
3. Memprogram digital I/O sebagai Running led dengan berbagai metode
4. Mengaplikasi fungsi digital I/O pada mikrokontroler AVR

II. ALAT DAN BAHAN :

1. Minsys ATMEGA8535
2. Downloader USBasp/ISP
3. AVR DUDE
4. CVAVR
5. Proteus
6. Jumper

III. DASAR TEORI

AVR dengan package 40-pin PDIP, contoh ATmega 8535 memiliki 32 I/O lines. Masing-masing lines dapat diatur fungsinya. Sebagian dari lines-lines I/O tersebut ada yang memilki fungsi-fungsi khusus, seperti ADC, Analog Comparator, PWM, USART dan External Interrupt.

Microcontroller AVR memiliki 2 register yang berhubungan dengan fungsi dasar ini, yaitu:

1. Register DDRx

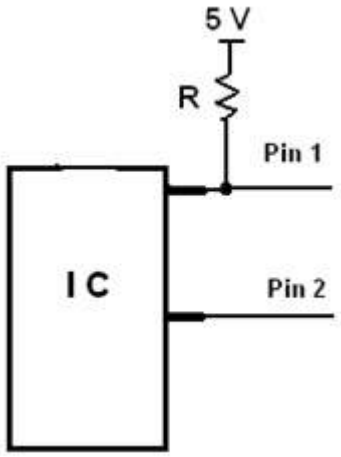
DDR***x*** (***x*** adalah port micro yang digunakan).
Misal: menggunakan Port A berarti registernya DDRA.
Register ini berfungsi untuk mengatur **arah** Pin/Port micro. Apakah dipakai sebagai input atau output. Nilai register ini sebesar **8 bit**. Setiap bit mewakili masing-masing pin kaki micro. Jika pin kaki micro digunakan sebagai **input** maka register **DDR*x*** nya harus **bernilai 0(nol)**. Jika pin kaki micro digunakan sebagai **output** maka register **DDR*x*** nya harus **bernilai 1**.

Misal Port A akan digunakan sebagai input maka untuk mensetnya kita gunakan perintah:
DDRA= 0x00; //ini berarti seluruh pin-pin pada Port A digunakan sebagai input
Contoh:
Port A.0 dijadikan input sedangkan Port A.2 ..sampai.. Port A.7 digunakan sebagai output maka perintahnya:
DDRA= 0xFE;
0xFE (heksa) == 0b 1111 1110 (biner).
Setiap bit nilai biner mewakili Pin-Pin pada PortA.

2. Register PORTx

PORT***x*** (***x*** adalah port micro yang digunakan). Misal: menggunakan Port A berarti registernya PORTA. **Jika Port digunakan sebagai input** register ini berfungsi sebagai penentu apakah kondisi Port di **Pull Up atau Floating**. **Jika Port digunakan sebagai output** register ini menentukan kondisi Port **High atau Low**.

Untuk memahami apa arti Pull Up perhatikan gambar berikut:



Pin 1. dihubungkan ke VCC(tegangan 5 Volt) melalui resistor (R), inilah yang di maksud dengan **Pull Up.** Saat tidak ada tegangan dari luar Pin 1 akan **cenderung berkondisi High (1)**

Pin 2. dibiarkan begitu saja sehingga kondisi logic Pin2 rentan terhadap pengaruh sekitarnya. Pin 2. bisa berlogika high bisa juga berlogika low ini artinya logika Pin2 mengambang (**Floating**). Kondisi floating biasanya diperlukan saat Pin sebuah IC atau micro dihubungkan ke sensor. Karena jika di Pull Up dikhawatirkan kondisi logic Pin IC mengganggu kondisi logic pin-pin sensor.

Perhatikan *code program* berikut:

```
DDRA=0x00;
PORTA = 0xFF;
```

ini berarti seluruh pin-pin pada Port A sebagai Input dan di Pull Up.

Perhatikan *code program* berikut:

```
DDRB= 0x00;
PORTB=0x0F;
0x0F == 0b 0000 1111
```

berarti :

Seluruh pin-pin PortB dijadikan Input

PortB.0 ..sampai.. PortB.3 di Pull Up

PortB.4 ..sampai.. PortB.7 dalam keadaan Floating

3. **Infinite Looping**

Infinite : tak terbatas, tak terhingga

Looping : perulangan, pengulangan, perputaran

Infinite Looping adalah perulangan (looping) yang dijalankan terus menerus. code program yang dijalankan AVR adalah code program yang berada dalam fungsi utama.

Program dijalankan berurutan dari atas ke bawah dan program tersebut hanya akan dijalankan **sekali**. Dengan menggunakan infinite looping *Code Program* yang berada di dalam infinite looping akan dijalankan terus menerus. Untuk **keluar dari infinite looping** digunakan perintah **break;**

IV. TUGAS PENDAHULUAN

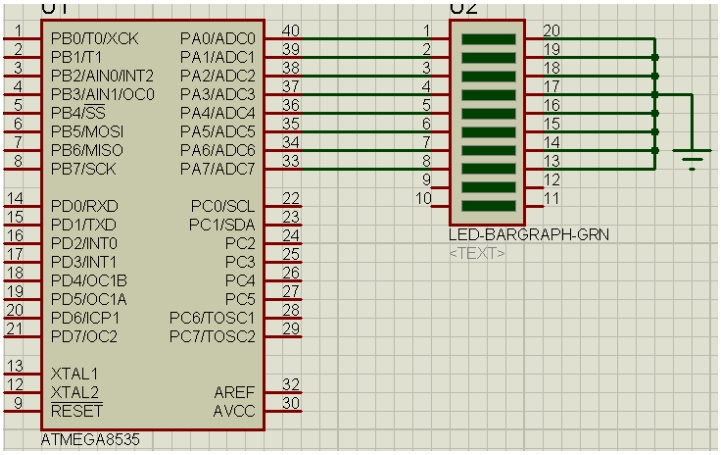
1. Sebutkan register-register dalam AVR
2. Apa yang dimaksud dengan Infinite dan Looping.
3. Buatkan code program fungsi utama.
4. Untuk keluar dari infinite looping digunakan perintah ?

V. PROSEDUR DAN PENGAMATAN PERCOBAAN

Metode Praktikum

1. Secara keseluruhan beberapa percobaan dibawah ini menggunakan cvavr sehingga kode digenerate secara otomatis.
2. Atur semua kode program yang ada kemudian diletakkan sesuai pada bagian-bagian kode hasil generate cvavr.
3. Pastikan hasil compile tidak menghasilkan error dengan menekan F9.
4. Gunakan proteus untuk melakukan simulasi
5. Selanjutnya download program tersebut ke Mikrokontroler
6. Amati dan analisis hasilnya kemudian catat hasil tersebut sebagai laporan sementara

A. Praktikum digital output



Gambar 1.

Prosedur Praktikum

Buatlah rangkaian seperti pada gambar 1.

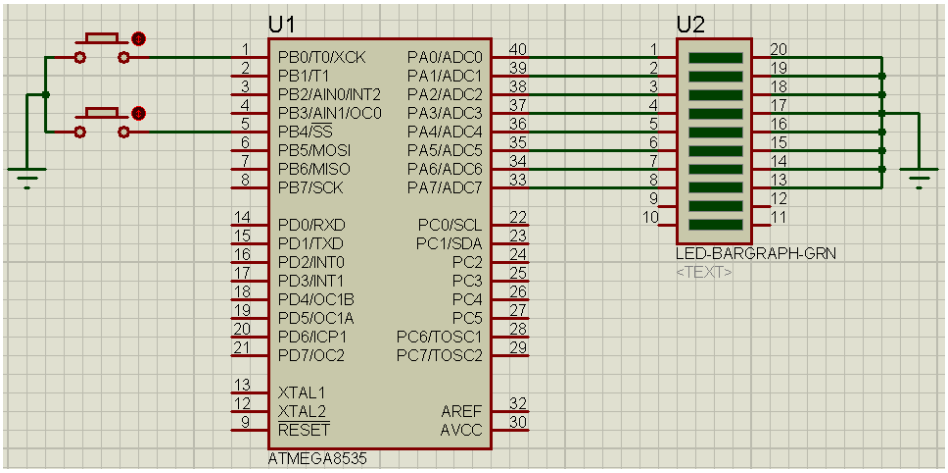
a. Program LED menyala semua secara bersama
b. Program LED Led-1 On, Led-2 Off, Led-3 On, Led-4 Off, Led-5 On, Led-6 Off, Led-7 On, Led-8
c. Program LED berkedip bersamaan
d. Program LED geser bergantian ke-kanan
e. Program LED geser bergantian ke-kekiri

B. Praktikum digital input-output

Prosedur Praktikum

Buatlah rangkaian seperti pada gambar 2.

a. Jika saklar SW0 ditekan (tertutup) LED0 akan menyala, dan sebaliknya.
b. Saklar Sw0 untuk menghidupkan LED, Saklar Sw1 untuk mematikan LED
c. Buatlah program apabila ditekan Sw1 ditekan nyala LED berjalan bergantian ke kiri, apabila ditekan Sw2 nyala LED berjalan bergantian ke kanan.



Gambar 2.

d. Buatlah program untuk menjalankan dua buah led nyala bergantian terus menerus dengan jeda pindah mendekati 2 detik.
e. Sw1 = sebagai START (mulai menjalankan led bergantian)
f. Sw2 = sebagai STOP (mengganti led berjalan satu)

MODUL VI
INTERUPSI

I. TUJUAN :

1. Mampu mengakses perangkat interupsi pada mikrokontroler AVR
2. Mengaplikasi perangkat interupsi pada mikrokontroler AVR

II. ALAT DAN BAHAN :

1. Minsys ATMEGA8535
2. Downloader USBasp/ISP
3. AVR DUDE
4. CVAVR
5. Proteus
6. Jumper

III. DASAR TEORI

Interupsi adalah terjadinya penyelaan terhadap program utama, yang mengharuskan program utama berhenti dulu dan mendahulukan interupsi. Interrupt akan menghentikan sementara eksekusi dari jalannya program yang normal. Apabila terjadi interrupt, maka eksekusi program akan menuju ke program interrupt, setelah selesai maka eksekusi program akan kembali ke program semula. Interupt sangat dibutuhkan dalam permrograman karena dengan interrupt kita dapat mengontrol suatu kejadian di luar system. Tanpa interrupt kita dapat mengecek suatu kondisi menggunkan looping, teknik ini dinamakan polling. Tapi polling memiliki beberapa kekurangan diantaranya kita tidak bisa melakukan pekerjaan lain selama kita melakukan pengecekan ataupun jika dimungkinkan program akan sangat panjang.

Jika terjadi suatu interrupt, maka eksekusi program akan menuju ke program interrupt yang ditunjukkan dengan vector interrupt. Setiap mikrokontroller memiliki beberapa fasilitas interrupt sehingga tiap-tiap interrupt memiliki vector interrupt yang berbeda-beda. Vector interrupt merupakan alamat dimana program interrupt harus ditempatkan agar dapat dieksekusi dengan benar. Umumnya alamat antar vector interuput hanya berjarak 1 byte sehingga apabila program interrupt yang dibuat sangat panjang akan mengganggu vector intruupt yang lain. Untuk mengatasi hal tersebut dalam pembuatan program akan dilompatkan ke label dimana program interrupt tersebut disediakan. Pada mikrokontroller ATMega8535 jenis-jenis interrupt beserta alamat vector interrupt dapat dilihat di datasheet ATMega8535

Vector No.	Program Address(2)	Source	Interrupt Definition
1	0x000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog Reset
2	0x001	INT0	External Interrupt Request 0
3	0x002	INT1	External Interrupt Request 1
4	0x003	TIMER2 COMP	Timer/Counter2 Compare Match
5	0x004	TIMER2 OVF	Timer/Counter2 Overflow
6	0x005	TIMER1 CAPT	Timer/Counter1 Capture Event
7	0x006	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	0x007	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	0x008	TIMER1 OVF	Timer/Counter1 Overflow
10	0x009	TIMER0 OVF	Timer/Counter0 Overflow
11	0x00A	SPI, STC	Serial Transfer Complete
12	0x00B	USART, RXC	USART, Rx Complete
13	0x00C	USART, UDRE	USART Data Register Empty
14	0x00D	USART, TXC	USART, Tx Complete
15	0x00E	ADC	ADC Conversion Complete
16	0x00F	EE_RDY	EEPROM Ready
17	0x010	ANA_COMP	Analog Comparator
18	0x011	TWI	Two-wire Serial Interface
19	0x012	INT2	External Interrupt Request 2
20	0x013	TIMER0 COMP	Timer/Counter0 Compare Match
21	0x014	SPM_RDY	Store Program Memory Ready

Alamat vector reset dan vector interrupt dapat dirubah-rubah sesuai dengan keinginan kita dengan cara mengubah BOOTRTS dan IVSEL. BOOTRST dapat dirubah melalui program downloader yang digunakan. Kalau menggunakan PonyProg2000 dapat melaui sekuriti konfigurasi.Sedangkan untuk mengkonfigurasi IVSEL dapat dilakukan pada regiser GICR. Adapun keterangan dari dua bit GICR adalah sebagai berikut:

BIT 1 – IVSEL ♉ Interrupt Vector Select
Ketika di set nol, interrupt vector akan ditempatkan di awal Flash memory. Jika di set satu, interrupt vector akan dipindahkan di awal Boot Loader dari flash. Alamat dari Boot Flash dapat ditentukan dari BOOTSZ
Fuses. Konfigurasi dari BOOTSZ dapat dilihat dalam tabel dibawah ini

BIT 0 – IVCE ♉ Interrupt Vector Enable
Bit ini harus di set satu agar perubahan pada IVSEL dapat diaktivkan. IVCE akan di nolkan oleh hardware untuk 4 cycle setelah IVSEL di tulis. Setting IVCE akan menonaktifkan interrupt. Ketika power off atau reset maka program akan menuju ke alamat $000 untuk menjalankan program reset.
Pada bahasa C program akan menuju ke fungsi main(). Pada C untuk dapat mengakses interrupt diperlukan sebuah library interrupt.h adapun penuisannya sebagai berikut :

```
#include <avr/interrupt.h>
```

Semua program interrupt dapat dideskripsikan menggunakan macro command ISR(), seperti

```
ISR(INT0_vect)
{
//program yang akan ditulis
}
```

IV. TUGAS PENDAHULUAN

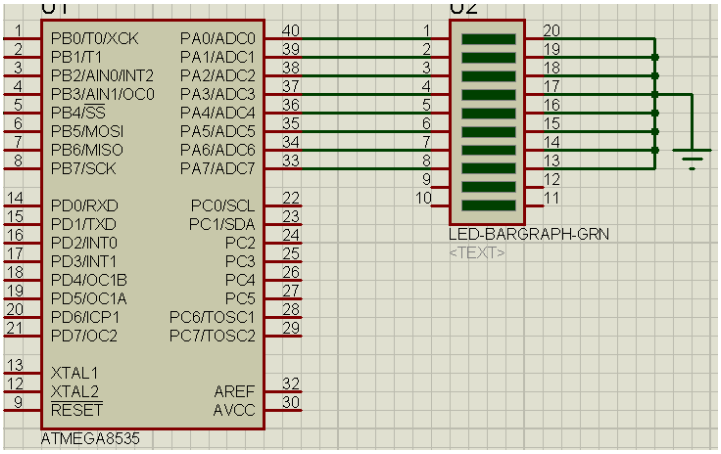
1. Sebutkan register-register dalam AVR
2. Apa yang dimaksud dengan Infinite dan Looping.
3. Buatkan code program fungsi utama.
4. Untuk keluar dari infinite looping digunakan perintah ?

V. PROSEDUR DAN PENGAMATAN PERCOBAAN

Metode Praktikum

1. Secara keseluruhan beberapa percobaan dibawah ini menggunakan cvavr sehingga kode digenerate secara otomatis.
2. Atur semua kode program yang ada kemudian diletakkan sesuai pada bagian-bagian kode hasil generate cvavr.
3. Pastikan hasil compile tidak menghasilkan error dengan menekan F9.
4. Gunakan proteus untuk melakukan simulasi
5. Selanjutnya download program tersebut ke Mikrokontroler
6. Amati dan analisis hasilnya kemudian catat hasil tersebut sebagai laporan sementara

A. Praktikum digital output



Gambar 1.

Prosedur Praktikum

Buatlah rangkaian seperti pada gambar 1.

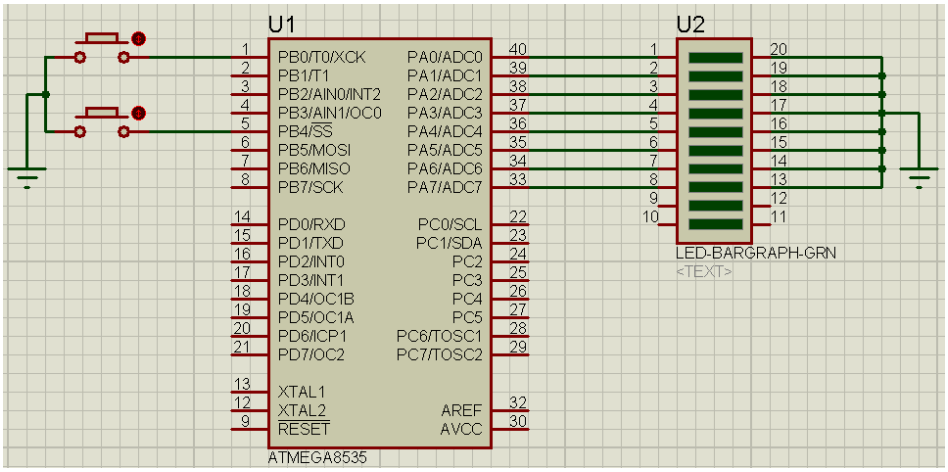
a. Program LED menyala semua secara bersama
b. Program LED Led-1 On, Led-2 Off, Led-3 On, Led-4 Off, Led-5 On, Led-6 Off, Led-7 On, Led-8
c. Program LED berkedip bersamaan
d. Program LED geser bergantian ke-kanan
e. Program LED geser bergantian ke-kekiri

B. Praktikum digital input-output

Prosedur Praktikum

Buatlah rangkaian seperti pada gambar 2.

a. Jika saklar SW0 ditekan (tertutup) LED0 akan menyala, dan sebaliknya.
b. Saklar Sw0 untuk menghidupkan LED, Saklar Sw1 untuk mematikan LED
c. Buatlah program apabila ditekan Sw1 ditekan nyala LED berjalan bergantian ke kiri, apabila ditekan Sw2 nyala LED berjalan bergantian ke kanan.



Gambar 2.

d. Buatlah program untuk menjalankan dua buah led nyala bergantian terus menerus dengan jeda pindah mendekati 2 detik.
e. Sw1 = sebagai START (mulai menjalankan led bergantian)
f. Sw2 = sebagai STOP (mengganti led berjalan satu)

MODUL VII
TIMER DAN COUNTER

I. TUJUAN :

1. Mahasiswa mampu menggunakan fitur timer/counter mikrokontroler.
2. Mahasiswa mampu menggunakan mikrokontroler untuk membuat timer.
1. Mahasiswa mampu menggunakan mikrokontroler untuk menghitung banyaknya pulsa yang masuk.

II. ALAT DAN BAHAN :

1. Minsys ATMEGA8535
2. Downloader USBasp/ISP
3. AVR DUDE
4. CVAVR
5. Proteus
6. Jumper

III. DASAR TEORI

Timer/counter dalam mikrokontroler merupakan fasilitas yang salah satu fungsinya sebagai pewaktu atau hitungan. Sumber clock atau trigger dapat dibangkitkan dari sinyal eksternal atau internal. Jika sumber sinyal berasal dari internal maka di sebut sebagai TIMER dan jika sumber sinyal berasal dari luar maka disebut sebagai COUNTER. Mikrokontroler ATMega memiliki 3 buah timer yaitu Timer0, Timer1, dan Timer2. Timer0 dan Timer2 memiliki kapasitas 8-bit sedangkan Timer1 memiliki kapasitas 16-bit.

Timer 8 bit adalah timer yg bisa mencacah/menghitung sampai maksimal nilai 0xFF heksa (dalam biner = 1111 1111). Klo yg 16 bit nilai maksimalnya 0xFFFF.

Pada penulisan ini akan dibahas mengenai timer1 16 bit. Untuk dapat menjalankan timer/counter1 harus dipelajari dulu mengenai register timer/counter1 karena di register itulah tempat setting Timer/Counter1 agar bisa bekerja. Register tersebut terdiri dari :

-TCCR1B
-TCNT1
-TIMSK
-TIFR.

Register TCCR1B merupakan tempat setting clock yang intinya agar timer/counter1 bisa bekerja maka register ini jangan sampai diisi dengan 0×00 (dikosongkan). Sumber clock bisa berasal dari internal mulai dari no prescaler sampai 1024 prescaler dan bisa juga dari sumber external.

Register TCNT1 merupakan register pencacah setiap ada trigger bisa tepi naik atau tepi turun, tapi kalo sumbernya dari dalam (internal) pencacahan dilakukan pada saat tepi naik. Register ini akan mencacah naik dari 0×00 sampai nilai max 0xFFFF kemudian di reset kembali lagi ke 0×00. Pada saat overflow yaitu kondisi dari 0xFFFF ke 0×00 maka bit TOV1 dari register TIFR akan di set 1. Keadaan overflow juga bisa digunakan untuk menjalankan interrupt.

Register TIMSK merupakan register tempat setting interrupt timer/counter1 overflow diaktifkan atau tidak. Dengan memberikan logika satu pada bit TOIE1 maka interrupt timer/counter1 aktif dengan catatan global interrupt diaktifkan (terdapat di Status register). Sedangkan register TIFR (Timer/Counter Interrupt Flag Register) digunakan sebagai penanda apakah sudah terjadi overflow. Pada timer/counter1 overflow ditandai dengan logika 1 pada bit TOV1 pada register ini. rumus yang digunakan adalah :

TCNT = (1+0xFFFF) - (waktu *(XTAL / prescaler))
waktu --> waktu yg kita inginkan
XTAL --> frekuensi xtal yg dipakai
prescaler --> nilai prescaler

Apa nilai prescaler itu?
Timer membutuhkan clock source. Biasanya clock source yg saya pakai adalah clock sistem (XTAL). Dan kita bisa menset besarnya nilai ini. Maximum sama dengan XTAL, minimum XTAL/1024. Sehingga nilai pembagi (1024) ini yg disebur nilai prescaler.
Macam2 nilai prescaler yg diijinkan: **1**, **8** , **64** , **256** , **1024**
Untuk mengubah nilai prescaler timer 1, kita harus merubah nilai register **TCCR1B bit 0...2**

Table 48. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

IV. TUGAS PENDAHULUAN

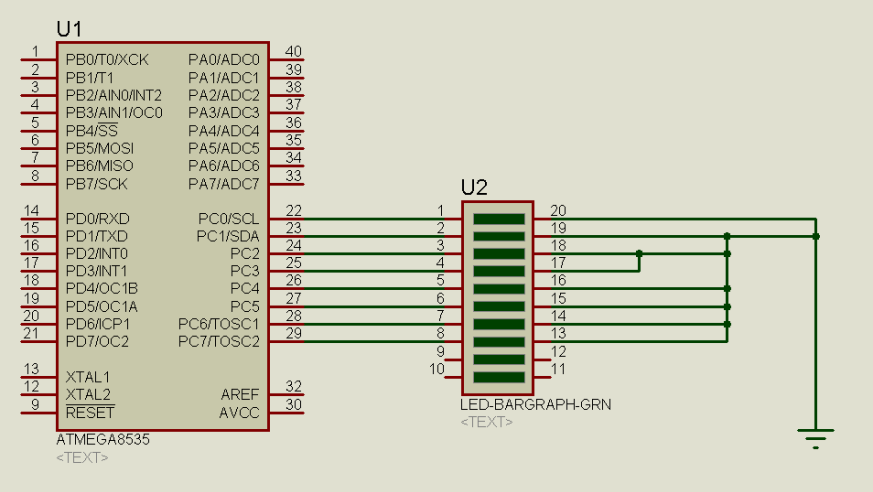
1. Apa yang dimaksud dengan Timer ?
2. Apa yang dimaksud dengan Counter.
3. Sebutkan register-register untuk tempat menyeting Timer/Counter ?.
4. Apa yang dimaksud nilai prescaler ?

V. PROSEDUR DAN PENGAMATAN PERCOBAAN

Metode Praktikum

1. Secara keseluruhan beberapa percobaan dibawah ini menggunakan cvavr sehingga kode digenerate secara otomatis.
2. Atur semua kode program yang ada kemudian diletakkan sesuai pada bagian-bagian kode hasil generate cvavr.
3. Pastikan hasil compile tidak menghasilkan error dengan menekan F9.
4. Gunakan proteus untuk melkukan simulasi
5. Selanjutnya download program tersebut ke Mikrokontroler
6. Amati dan analisis hasilnya kemudian catat hasil tersebut sebagai laporan sementara.

A. Praktikum Timer



Gambar 1.

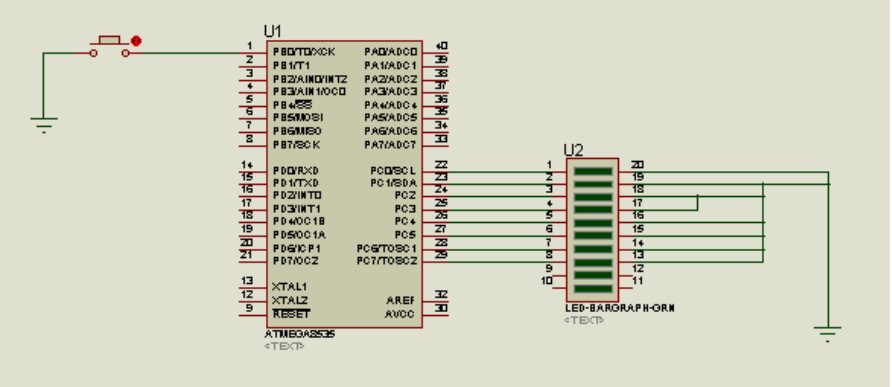
Buatlah rangkaian seperti pada gambar 1.
Buatlah aplikasi TIMER0 sehingga berakibat lampu pada PORTC berkedip-kedip dengan ketentuan, frekuensi osilator yang digunakan adalah 8 MHz, skala clocknya adalah 1024, clock value nya adalah 0.
Ketiklah :

```
#include <mega8535.h>
unsigned char led = 0xff;
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
TCNT0=0x00;
led =~led;
PORTC = led;
}
void main(void)
{
PORTC=0xFF;
DDRC=0xFF;
TCCR0=0x05;
TCNT0=0x00;
OCR0=0x00;
TIMSK=0x01;
#asm("sei")
while (1) ;
}
```

1. Buat pada timer0 untuk waktu delaynya 1 dt.
2. Buat pada timer1 dan timer 2 untuk soal 3.1.

B. Praktikum counter.

Buatlah rangkaian seperti Gambar 2



Gambar 2

Tuliskan :

```
#include <mega8535.h>
unsigned char led,a;
void inisialisasiTIMER () ;
void main(void)
{
PORTB=0x01;
DDRB=0x00;
led=0xFF;
DDRC=0xFF;

TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;
TIMSK=0x00;

inisialisasiTIMER ();
while (1)
    {
    a = TCNT0;
    if (a ==0x03)
{
    led = PINC;
    PORTC =~led;
    TCNT0 = 0x00;
    }

    }
}
 void inisialisasiTIMER ()
{
TCNT0 = 0x00;
TCCR0 = 0x07;
}
```

Jalankan program, tekan push button sebanyak 3 kali. Amati yang terjadi dan simpulkan.

MODUL VIII
LIQUID CRISTAL DISPLAY (LCD)

I. TUJUAN :

Mahasiswa mampu menggunakan LCD sebagai penampilan yang menggunakan kristal cair, sebagai tampilan data, baik karakter, huruf ataupun grafik.

II. ALAT DAN BAHAN :

1. Minsys ATMEGA8535
2. Downloader USBasp/ISP
3. AVR DUDE
4. CVAVR
5. Proteus
6. LCD 16 x 2
7. Jumper

III. DASAR TEORI

Modul LCD Character dapat dengan mudah dihubungkan dengan mikrokontroller. LCD yang akan kita pelajari bersama ini mempunyai lebar display 2 baris 16 kolom atau biasa disebut sebagai LCD Character 16x2, dengan 16 pin konektor, yang didefinisikan sebagai berikut:

Tabel 1 Pin dan Fungsi

PIN	Nama	Fungsi
1	VSS	Ground voltage
2	VCC	+5V
3	VEE	Contrast voltage
4	RS	Register Select 0 = Instruction Register 1 = Data Register
5	R/W	Read/ Write, to choose write or read mode 0 = write mode 1 = read mode
6	E	Enable 0 = start to latch data to LCD character 1= disable
7-14	DB0 – DB7	Data Bus
15	BPL	Back Plane Light
16	GND	Ground voltage

Display karakter pada LCD diatur oleh pin EN, RS dan RW: Jalur EN dinamakan Enable. Jalur ini digunakan untuk memberitahu LCD bahwa anda sedang mengirimkan sebuah data. Untuk mengirimkan data ke LCD, maka melalui program EN harus dibuat logika low "0" dan set pada dua jalur kontrol yang lain RS dan RW. Ketika dua jalur yang lain telah siap, set EN dengan logika "1" dan tunggu untuk sejumlah waktu tertentu (sesuai dengan datasheet dari LCD tersebut) dan berikutnya set EN ke logika low "0" lagi. Jalur RS adalah jalur Register Select. Ketika RS berlogika low "0", data akan dianggap sebagi sebua perintah atau instruksi khusus (seperti clear screen, posisi kursor dll). Ketika RS berlogika high "1", data yang dikirim adalah data text yang akan ditampilkan pada display LCD.

Sebagai contoh, untuk menampilkan huruf "T" pada layar LCD maka RS harus diset logika high "1". Jalur RW adalah jalur kontrol Read/ Write. Ketika RW berlogika low (0), maka informasi pada bus data akan dituliskan pada layar LCD. Ketika RW berlogika high "1", maka program akan melakukan pembacaan memori dari LCD. Sedangkan pada aplikasi umum pin RW selalu diberi logika low "0". Pada akhirnya, bus data terdiri dari 4 atau 8 jalur (bergantung pada mode operasi yang dipilih oleh user). Pada kasus bus data 8 bit, jalur diacukan sebagai DB0 s/d DB7.

IV. TUGAS PENDAHULUAN

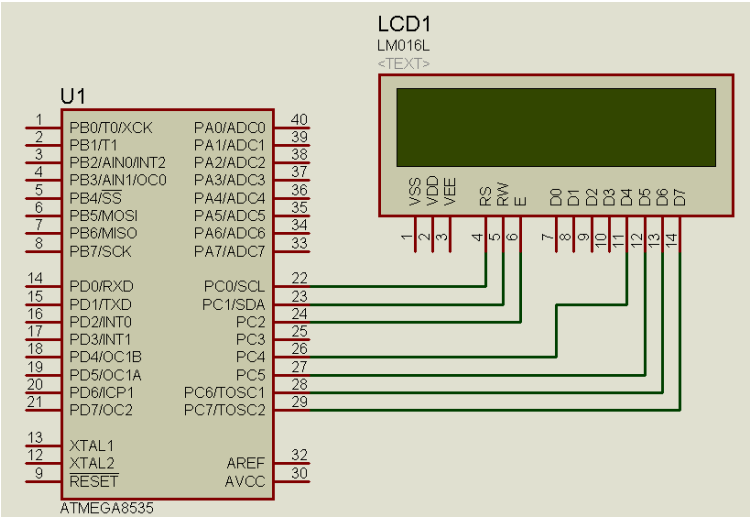
1. Apa fungsi dari LCD ?
2. Bagaimana mensetting pin-pin LCD.

V. PROSEDUR DAN PENGAMATAN PERCOBAAN

Metode Praktikum

1. Secara keseluruhan beberapa percobaan dibawah ini menggunakan cvavr sehingga kode digenerate secara otomatis.
2. Atur semua kode program yang ada kemudian diletakkan sesuai pada bagian-bagian kode hasil generate cvavr.
3. Pastikan hasil compile tidak menghasilkan error dengan menekan F9.
4. Gunakan proteus untuk melakukan simulasi
5. Selanjutnya download program tersebut ke Mikrokontroler
6. Amati dan analisis hasilnya kemudian catat hasil tersebut sebagai laporan sementara.

A. Praktikum Timer



Gambar 1.

Buatlah rangkaian seperti pada gambar 1.
Buatlah aplikasi TIMER0 sehingga berakibat lampu pada PORTC berkedip-kedip dengan ketentuan, frekuensi osilator yang digunakan adalah 8 MHz, skala clocknya adalah 1024, clock value nya adalah 0.

Ketiklah :

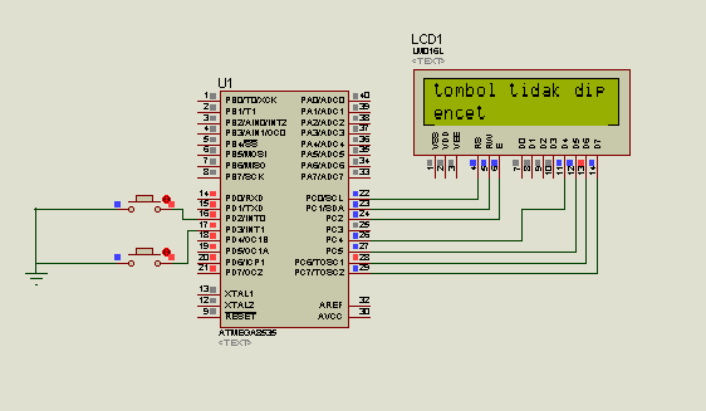
```
#include <mega8535.h>
#include <alcd.h>

void main(void)
{
        PORTC=0x00;
        DDRC=0x00;
        lcd_init(16);
while (1)
        {
            lcd_gotoxy(0,0);
            lcd_putsf("STMKG");
            lcd_gotoxy(0,1);
            lcd_putsf("Jaya");
         }
}
```

1. Buat angka dengan kenaikan 1 setiap 1 dt setiang hitungan 10 kembali ke hitungan 0;
2. Buatlah tulisan berjalan.

B. Praktikum counter.

Buatlah rangkaian seperti Gambar 2



Gambar 2.

1. Buatlah teks jika tombol 1 di tekan LCD menulis “tombol 1 ditekan", demikian juga tombol 2. Jika tidak ditekan maka teks “tombol tidak ditekan” .
2. Pada latihan 3.2, tambahkan external interrup, jika di tekan maka LCD tertulis “maaf saya ganngu 1 dt”.

MODUL IX
ANALOG DIGITAL CONVERTER (ADC)

I. TUJUAN :

1. Mengetahui dan memahami bagaimana memrogram mikrokontroler untuk mengonversi data analog menjadi data digital.
2. Mengetahui dan memahami cara menggunakan ADC yang ada di dalam mikrokontroler.

.

II. ALAT DAN BAHAN :

1. Minsys ATMEGA8535
2. Downloader USBasp/ISP
3. AVR DUDE
4. CVAVR
5. Proteus
6. LCD 16 x 2
7. Adaptor Variable.
8. Jumper

III. DASAR TEORI

ADC (Analog to Digital Converter) adalah fitur paling populer dari ATmega. Dengan adanya fitur ini kita tidak perlu menggunakan ADC0804 untuk membaca sinyal analog. ATmega8535 memiliki 8 channel input ADC. Hasil pembacaan ADC beresolusi maksimum 10 bit.

Register-Register yang mempengaruhi ADC:

Berikut adalah daftar register untuk menentukan setting ADC

Register ADMUX

Register ADMUX digunakan untuk menentukan tegangan referensi dari ADC menentukan format data hasil konversi ADC menentukan channel ADC yg akan digunakan

Berikut isi dari register ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	**REFS0**	**ADLAR**	**MUX4**	**MUX3**	**MUX2**	**MUX1**	**MUX0**	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7 (REFS1) dan bit 6(REFS0) digunakan untuk menentukan tegangan referensi ADC.

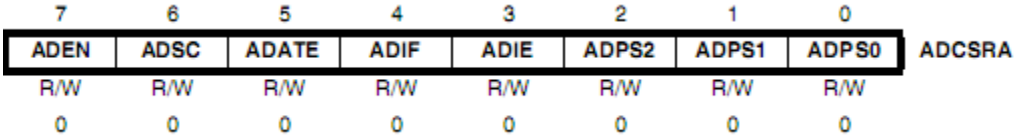
REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

Bit 5 (ADLAR) digunakan untuk menentukan format data hasil konversi.

Bit 3..0 (MUX3..0) digunakan untuk menentukan channel ADC

MUX 4..0	SINGGLE ENDED INPUT
00000	ADC0
00001	ADC1
00010	ADC2
00011	ADC3
00100	ADC4
00101	ADC5
00110	ADC6
00111	ADC7

Register ADCSRA



Bit 7 (ADEN) untuk mengaktifkan ADC. ADEN=0 disable / ADEN=1 enable
Bit 6 (ADSC) untuk memulai (start) pembacaan ADC.
Bit 5 (ADFR) jika ADFR=1 free running mode , ADFR=0 single conversion
Bit 4 (ADIF) bit penanda interupsi. Bernilai 1 saat konversi ADC selesai.
Bit 3 (ADIE) berfungsi untuk mengaktifkan interupsi ADC. ADIE=1 enable / ADIE=0 disable
Bit 2..0(ADPS2..0) menentukan *clock* ADC

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Register ADCL dan ADCH

Merupakan 2 register tempat menampung hasil pembacaan ADC. Untuk mengambil nilai nya gunakan ADCW(mode 10 bit) dan ADCH (mode 8 bit)

Berikut listing fungsi baca adc

```
unsigned int getadc(unsigned char channeladc)
    {
        unsigned int adcVal;
        ADMUX=channeladc|0x40;   //avcc
```

```
        ADCSRA|=(1<<ADEN);
        ADCSRA|=(1<<ADSC);
        loop_until_bit_is_clear(ADCSRA,ADSC);
        adcVal = ADCW;
        ADCSRA&=~(1<<ADEN);
        return adcVal;
    }
```

IV. TUGAS PENDAHULUAN

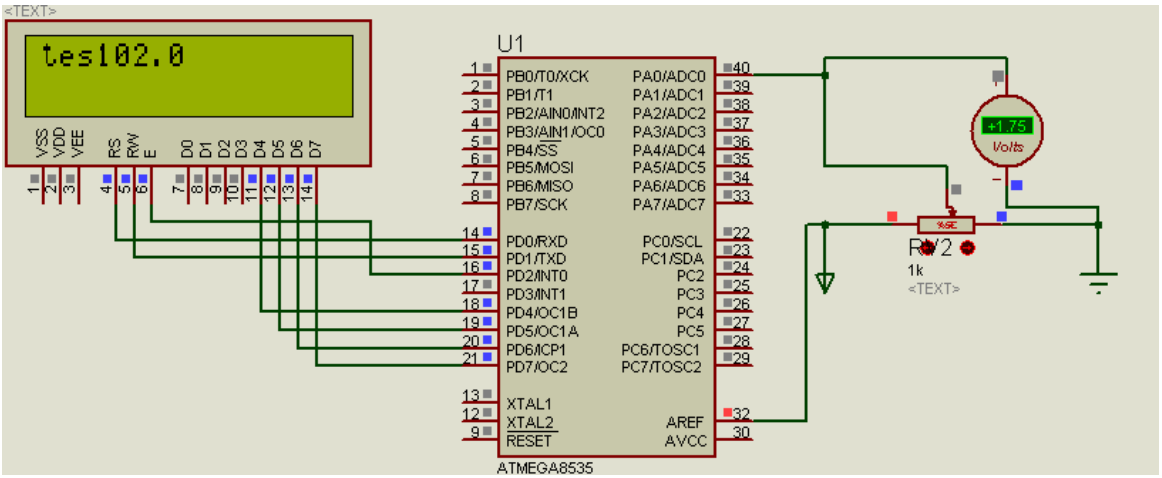
1. Apa yang dimaksud resolusi pada ADC ?
2. Berapa resolusi pada ATMEGA8535 ?
3. Sebutkan register-register yang digunakan untuk mengakses ADC pada ATMEGA8535.
4. Sebutkan listing untuk mengakses ADC.

V. PROSEDUR DAN PENGAMATAN PERCOBAAN

Metode Praktikum

1. Secara keseluruhan beberapa percobaan dibawah ini menggunakan cvavr sehingga kode digenerate secara otomatis.
2. Atur semua kode program yang ada kemudian diletakkan sesuai pada bagian-bagian kode hasil generate cvavr.
3. Pastikan hasil compile tidak menghasilkan error dengan menekan F9.
4. Gunakan proteus untuk melakukan simulasi
5. Selanjutnya download program tersebut ke Mikrokontroler
6. Amati dan analisis hasilnya kemudian catat hasil tersebut sebagai laporan sementara.

Praktikum ADC



Gambar 1.

1. Buatlah aplikasi ADC 10 bits dengan inputan analognya menggunakan PORTA.0 dan LCD ke PORTD seperti pada Gambar1. Rubah potensiometer sehingga keluarannya adalah sesuai yang tertera dengan Voltmeter DC dan pada LCD.
2. Ulangi percobaan 3.1 untuk ADC 8 bits.
3. Buat aplikasi adc untuk pengukuran suhu dengan sensor LM35, buat untuk adc 8 bits dan 10 bits.

MODUL X
KOMUNIKASI SERIAL

I. TUJUAN :

1. Mengetahui dan mengerti cara melakukan komunikasi serial dengan mikrokontroller AVR untuk mengendalikan sesuatu peralatan
2. Mampu menerapkan bahasa pemrograman C pada mikrokontroller AVR untuk melakukan komunikasi serial.

.

II. ALAT DAN BAHAN :

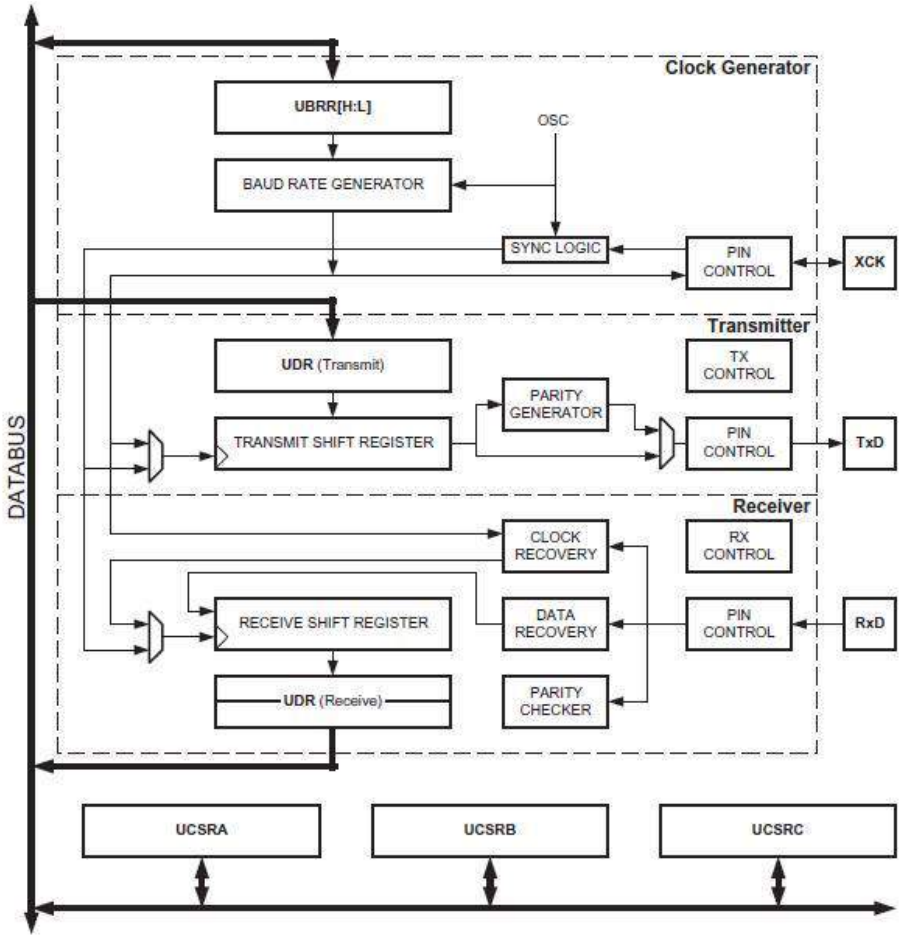
1. Minsys ATMEGA8535, dua buah
2. Downloader USBasp/ISP
3. AVR DUDE
4. CVAVR
5. Proteus
6. Sensor LM35
7. LCD 16 x 2
8. Adaptor Variable.
9. Jumper

III. DASAR TEORI

Komunikasi serial pada mikrokontroler AVR menggunakan fasilitas **USART** singkatan dari ***Universal Syncronous and Asyncronous Receiver Transmitter*** sangat handal dan berguna dalam berbagai aplikasi yang berhubungan antarmuka komunikasi serial dengan PC atau sesama mikrokontroler AVR atau bahkan mikrokontroler lain yang memiliki fasilitas komunikasi serial, misalnya program pemantauan suhu ruangan sekaligus pengontrolan AC atau kipas menggunakan antarmuka program PC Visual Basic atau Delphi, dan lain-lain. Jalur komunikasi serial pada mikrokontroler AVR ATMega16/32/8515/8535 terdapat pada pin PORTD.0 (RXD) dan PORTD.1 (TXD). Diagram bloknya ditunjukkan pada Gambar 1.

Berikut fitur-fitur komunikasi serial atau USART pada ATMega16:

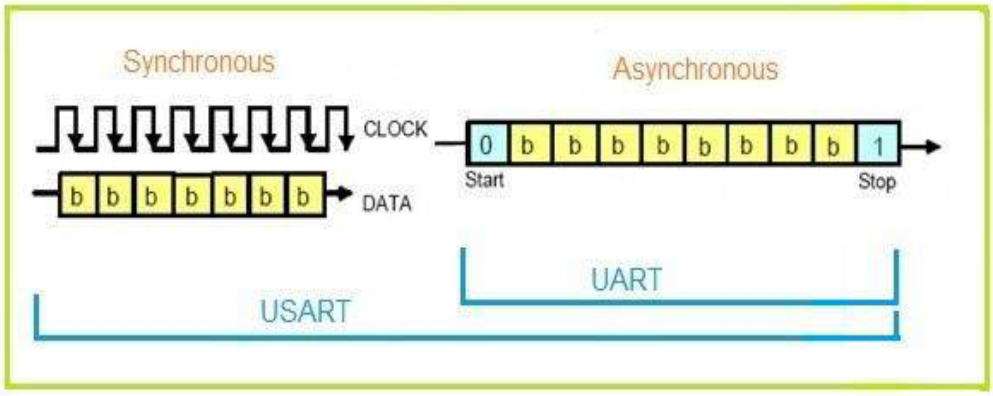
- *Full Duplex Operation (Independent Serial Receive and Transmit Registers)*
- *Asynchronous or Synchronous Operation*
- *Master or Slave Clocked Synchronous Operation*
- *High Resolution Baud Rate Generator*
- *Supports Serial Frames with 5, 6, 7, 8, or 9 Data Bits and 1 or 2 Stop Bits*
- *Odd or Even Parity Generation and Parity Check Supported by Hardware*
- *Data OverRun Detection*
- *Framing Error Detection*
- *Noise Filtering Includes False Start Bit Detection and Digital Low Pass Filter*
- *Three Separate Interrupts on TX Complete, TX Data Register Empty, and RX Complete*
- *Multi-processor Communication Mode*
- *Double Speed Asynchronous Communication Mode*



Gambar 1. Diagram blok fasilitas komunikasi serial AVR

Komunikasi dapat digunakan dengan 2 Mode :

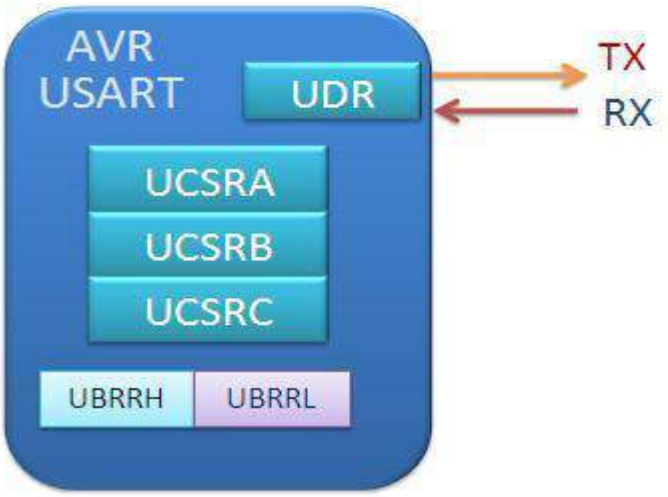
1. mode sinkron dimana pengirim data mengeluarkan pulsa/clock untuk sinkronisasi data.
2. mode asinkron, dimana pengirim data tidak mengeluarkan pulsa/clock, tetapi untuk proses sinkronisasi memerlukan inisialisasi, agar data yang diterima sama dengan data yang dikirimkan. Pada proses inisialisasi ini setiap perangkat yang terhubung harus memiliki baud rate (laju data) yang sama



Gambar 2. Mode Komunikasi

Mikrokontroler Atmega8535 memiliki *register* I/O yang berkaitan dengan komunikasi serial memakai UART

- Register Data (UDR), menyimpan data yg dikirim dan diterima.
- Register Control (UCSRA bit 0- bit1, UCSRB dan UCSRC)
  UCSRB digunakan untuk mengaktifkan penerimaan dan pengiriman data USART
  UCSRC (USART Control and Status Register C) digunakan untuk mengatur mode komunikasi USART
- Register Status (UCSRA bit 2 - bit 7)



Gambar 3. Mode Komunikasi

UBRR (*USART Baud Rate Register*) digunakan untuk menentukan baud rate

Operating Mode	Equation for Calculating Baud Rate	Equation for Calculating UBBR Value
Asynchronous Normal Mode (U2X=0)	$BAUD = \frac{f_{osc}}{16(UBBR+1)}$	$UBBR = \frac{f_{osc}}{16BAUD} - 1$
Asynchronous Double Speed Mode (U2X=0)	$BAUD = \frac{f_{osc}}{8(UBBR+1)}$	$UBBR = \frac{f_{osc}}{8BAUD} - 1$
Asynchronous Master Mode	$BAUD = \frac{f_{osc}}{2(UBBR+1)}$	$UBBR = \frac{f_{osc}}{2BAUD} - 1$

Instruksi dalam komunikasi serial.

printf("……..");
Digunakan untuk mencetak atau mengeluarkan data string (kata/kalimat) ke jalur komunikasi serial mikrokontroler
printf("%d %s %d", day, monthname, year);
putchar('…'); atau putchar(no_char)
Digunakan untuk mencetak atau mengeluarkan data char (karakter) ke jaluar komunikasi serial mikrokontroler
scanf(&varibel_penyimpan);
Digunakan untuk membaca/menerima data string/char dari jalur komunikasi serial mikrokontroler
scanf("%d %s %d", &day, monthname, &year);

getchar();
Digunakan untuk membaca/menerima data char (karakter) dari jalur komunikasi serial mikrokontroler
\r\n untuk enter dan newline
Menerima data serial bisa menggunakan fungsi **getchar() dan** fungsi **scanf**. Fungsi scanf sendiri ada di library **stdio.h, misalkan**

```
scanf("%d %s %d", &day, monthname, &year);  //terima data UART
//-------kirim data ke LCD--------
lcd_clear();
lcd_gotoxy(0,0);
sprintf(kata,"%d %s %d", day, monthname, year);
lcd_puts(kata);
printf("%d %s %d", day, monthname, year);   //kirim data UART
itoa(i,data_string)//konversi dari integer ke string
puts(data_string);//kirim data string
```

IV. TUGAS PENDAHULUAN

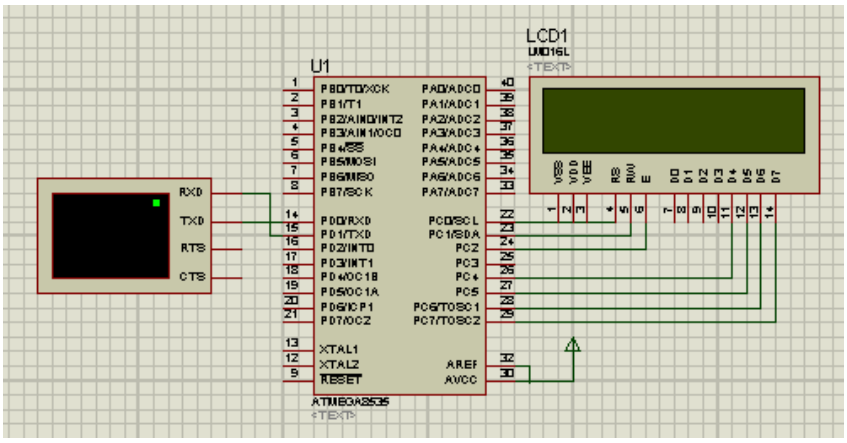
1. Apa yang dimaksud dengan Clock Generator ?
2. Apa yang dimaksud dengan Transmitter ?
3. Apa yang dimaksud dengan Receiver ?
4. Sebutkan register I/O yang digunakan dalam Komunikasi Serial USART.

V. PROSEDUR DAN PENGAMATAN PERCOBAAN

Metode Praktikum

1. Secara keseluruhan beberapa percobaan dibawah ini menggunakan cvavr sehingga kode digenerate secara otomatis.
2. Atur semua kode program yang ada kemudian diletakkan sesuai pada bagian-bagian kode hasil generate cvavr.
3. Pastikan hasil compile tidak menghasilkan error dengan menekan F9.
4. Gunakan proteus untuk melakukan simulasi
5. Selanjutnya download program tersebut ke Mikrokontroler
6. Amati dan analisis hasilnya kemudian catat hasil tersebut sebagai laporan sementara.

A. Praktikum Menampilkan Karakter Pada LCD



Gambar 4. Menampilkan Karakter

1. Buatlah aplikasi pengiriman data serial, kemudian tampilkan tulisan " STMKG OK"
2. Tampilkan poin 1 ke LCD.
3. Jika ditekan huruf H pada keyboard, LCD dan serial menampilkan tulisan "Anda menekan huruf H".

B. Praktikum Dengan Inputan LM35.

1. Buatlah aplikasi pengiriman data serial dan tampilan LCD dengan inputan berupa sensor suhu LM35 dan sensor cahaya.
2. Tambahkan 3 push button, jika push button 1 ditekan maka yang muncul pada serial dan LCD data sensor suhu dan Cahaya
3. Jika push button 2 ditekan maka yang muncul pada serial dan LCD data sensor suhu.
4. Jika push button 3 ditekan maka yang muncul pada serial dan LCD data Cahaya Tampilkan.

C. Praktikum Dengan dua Mikroprosesor.

1. Buatlah aplikasi pengiriman data serial antara 2 buah mikon. Dimana satu pada praktikum 2 dihubungkan Tx nya dengan Rx mikon 2.
2. Pada mikon 2 tambahkan LCD dan virtual terminal.
3. Tampilkan hasil di mikon 1 sama dengan mikon 2.

MODUL XI
INPUT SENSOR DIGITAL DHT11

I. TUJUAN :

1. Mengetahui dan mengerti cara menggunakan mikrokontroller AVR untuk inputan sensor l digital.
2. Mampu menerapkan bahasa pemrograman C pada mikrokontroller AVR untuk melakukan inputan keluaran digital dari sensor.

.

II. ALAT DAN BAHAN :

1. Minsys ATMEGA8535
2. Downloader USBasp/ISP
3. AVR DUDE
4. CVAVR
5. Proteus
6. Sensor DHT11, tiga buah
7. LCD 16 x 2
8. Adaptor Variable.
9. Jumper

III. DASAR TEORI

Sensor DHT11.

Sensor DHT11 merupakan sensor suhu yang memiliki keluaran yang berupa sinyal digital dengan konversi dan perhitungan dilakukan oleh mikrokontroler dan sudah terkalibrasi secara akurat dengan kompensasi suhu di ruang penyesuaian dengan nilai koefisien kalibrasi yang lebih akurat dan presisi

Prinsip kerja sensor ini yaitu menghitung perubahan nilai tahanan. Perubahan nilai tahanan dipengaruhi oleh perubahan suhu yang mengenai termistor. Termistor yang digunakan yaitu jenis *Negative Temperature Coefficient* (NTC) yang memiliki koefisien negative. Nilai resistansi akan turun jika suhu di sekitar NTC tinggi.

IV. TUGAS PENDAHULUAN

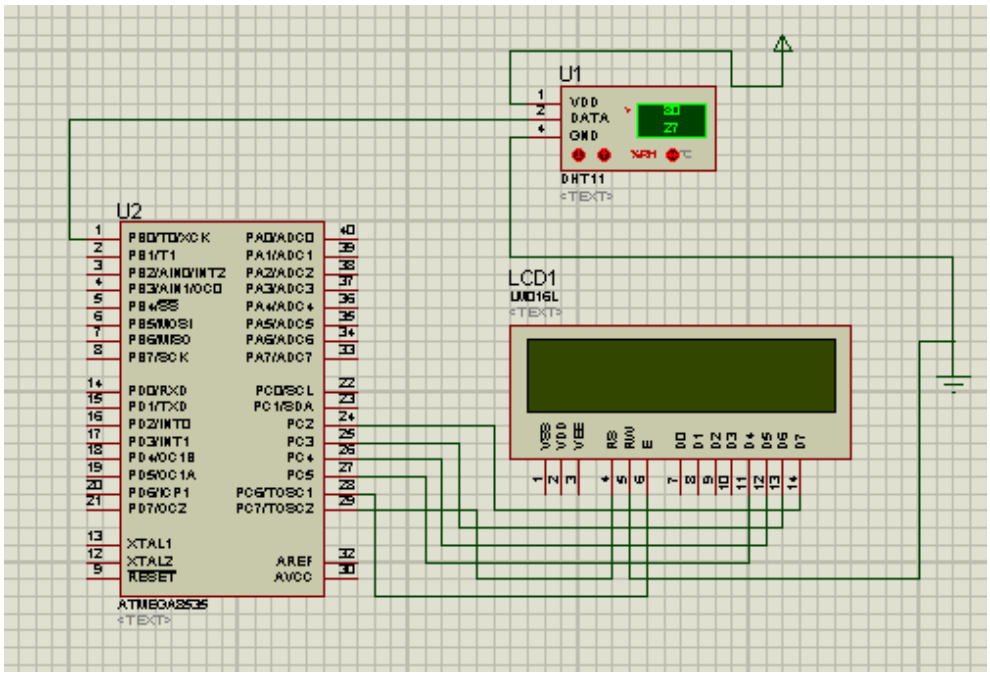
1. Bagaimana prinsip kerja Sensor DHT11 ?
2. Data apa saja yang dihasilkan Sensor DHT11 ?
3. Bagaimana mengatur pin DHT11 pada Mikroprosesor ATMEGA8535 ?

V. PROSEDUR DAN PENGAMATAN PERCOBAAN

Metode Praktikum

1. Secara keseluruhan beberapa percobaan dibawah ini menggunakan cvavr sehingga kode digenerate secara otomatis.
2. Atur semua kode program yang ada kemudian diletakkan sesuai pada bagian-bagian kode hasil generate cvavr.
3. Pastikan hasil compile tidak menghasilkan error dengan menekan F9.
4. Gunakan proteus untuk melakukan simulasi
5. Selanjutnya download program tersebut ke Mikrokontroler
6. Amati dan analisis hasilnya kemudian catat hasil tersebut sebagai laporan sementara.

A. Praktikum Dengan 1 Inputan Sensor DHT11



Gambar 1.

1. Buatlah aplikasi input sensor DHT11.
2. Tampilkan poin 1 ke LCD.
3. Isikan file .hex kesimulasi proteus dan mikon.

B. Praktikum Dengan 2 Inputan Sensor DHT11

Ulangi praktikum 1 dengan menambahkan komunikasi serial.

C. Praktikum Dengan 3 Inputan Sensor DHT11

1. Ulangi praktikum 2 kemudian tambahkan sensor DHT11 menjadi 2 buah
2. Ulangi langkah no. 1 dengan menambahkan sensor DHT11 menjadi 3 buah.